March/April 1982 Volume 2, Number 2 $2.95 (USA) £1.50 (UK)

The magazine for Sinclair users

SYNC®

The Making of a Computer Game: Flattop Lander

Reliable Cassette Loading

Six Shooter Game

ZX Video Display: How it Works

Graphic Plots With Your ZX81

Horizontal Scrolling

The Sinclair ZX-81 is innovative and powerful. Now there's a magazine to help you get the most out of it.

GET IN SYNC

Thousands of smart consumers have picked the Sinclair ZX-81 as their personal computer. And, unlike many of today's bargains, this one can really give you your money's worth. Or it can turn into nothing but an expensive calculator. A Sinclair owner can putter along in first gear, missing the power and potential of the ZX-81, or he can shift into high, pushing the ZX-81 beyond imaginable limits. That's why thousands of smart consumers have picked **SYNC** as their computer magazine.

Right on Target

The ZX-81 is unique. There is nothing like it, nothing that comes close to packing so much power and versatility into one small package. Some computer magazines might publish one or two articles about the Sinclair each year, some never mention it. **SYNC** covers only the ZX-81 and its predecessor, the ZX-80. If an article doesn't apply to the Sinclair, if a game doesn't work on the Sinclair, you won't see it in **SYNC**. Our staff and contributors are Sinclair owners. Some started out as experts. Others started as readers and became experts.

How can a whole magazine find enough material about one small computer? By covering everything from hardware to software, by offering both new applications and old tricks with a new twist. Did you know that the Sinclair can generate music? Our readers found that out when we published a program and article showing how to do it, and explaining why it works. Do you know where to buy software, books, or peripherals for the ZX-81? We list resources in every issue, along with addresses for user's groups so you can get in touch with other Sinclair owners. But knowing where to buy is not enough by a long shot. And that's where we can really help you out.

Hard-Hitting Evaluations

As a Sinclair owner, you know the value of a dollar. But it isn't always easy to know the value of all the extras on the market. Face it, some programs are great, some aren't worth the tape they're stored on. We receive every new product for the Sinclair as soon as it is available, often months before it is on the market. And those products are reviewed and tested with a very critical eye. If an advertiser doesn't care for this sort of honesty, we don't care for his business. We haven't gotten where we are by patting backs, we've gotten there by giving the Sinclair owner the information he needs. But there's more to **SYNC** than just reviews.

Applications and Explanations

The ZX-81 comes with a very powerful Basic language. But power doesn't imply difficulty. We show you how to get the most from your computer, whether you want to write a game or keep track of a mailing list. And we don't stop with Basic. The Sinclair can be programmed in machine language. For the newcomer, we have articles explaining machine language from the ground up. For the old pro (and anyone who has been reading **SYNC** for a while will soon find himself in this category) we have sophisticated routines for animation, data handling, and every other aspect of programming.

Don't run your computer in first gear.

Topping if off, hardware articles cover everything from attaching a full-size keyboard to adding a tape monitor. Whether you are interested in software or soldering, we'll keep you busy. But we also know how to have fun.

Games of Every Kind

If you like to shoot down attacking spaceships, fight monsters in a dungeon, or land on the moon, we've got what you want. Every issue of **SYNC** is packed with games. There are classic computer games converted for the Sinclair, and new games designed specifically to exploit the capabilities of the ZX-81. Our contributors keep getting better and better, but that's not surprising, because the games come complete with tips and explanations. Programming tricks and special techniques are fully explained, so you can use them in your own games. We don't believe in keeping secrets.

SYNC is a Creative Computing publication. **Creative Computing** is the number 1 magazine of software and applications with over 150,000 circulation. The two most popular computer games books in the world, *Basic Computer Games* and *More Basic Computer Games* (combined sales over 500,000) are published by Creative Computing. Creative Computing Software manufactures over 150 software packages for six different personal computers.

Order SYNC Today and Save Money!

SYNC

P.O. Box 789-M
Morristown, N.J. 07960

YES! Send me *SYNC* for the term checked:

- ☐ One year (6 issues) $12.97—I save 19%!
- ☐ Two years (12 issues) $22.97—I save 28%!
- ☐ Three years (18 issues) $31.97—I save 33%!

Savings based on full one-year subscription price of $16.

CHECK ONE:

- ☐ Payment enclosed.
- ☐ Bill me later.

Mr.
Mrs.
Ms._____
(please print full name)

Address_____Apt.___

City_____

State_____Zip_____

Foreign postage: Add $3 a year for Canada. Add $5 a year (cash payment in U.S. currency only) for all other countries outside U.S. and possessions. Please allow 60 to 90 days for delivery of first issue.

8H005

The magazine for Sinclair users

SYNC

March/April 1982 Volume 2, Number 2

DEPARTMENTS

4 **Letters**

6 **SYNC Notes** ..... *Wren-Hilton*
The Second ZX Microfair Report

8 **Perceptions** ..... *Ornstein*
The ZX80/81 Video Display System

14 **Kitchen SYNC** ..... *Groupe, Tardiff, Zatkovich*
Using Fun Features to Make a Great Toy

39 **Try This** ..... *Phillips, Booth*

39 **Glitchoidz Report**

16 **Puzzles and Problems** ..... *Townsend*

48 **Resources**

MATH AND GRAPHICS

18 **Understanding Floating Point Arithmetic** ..... *Logan*
Part 2—The CALCULATOR Language

26 **Plotting with 4K Basic** ..... *Brendel*
Getting around PLOT, PRINT AT, and VAL in 4K

MACHINE LANGUAGE PROGRAMMING

31 **LSCROLL—For Spectacular Screen Displays** .... *Sharp*
A lateral scrolling program

35 **How to Invent a Game** ..... *Bobst*
Part 2 on using Basic and MC techniques

37 **Safe Machine Code Routines** ..... *Doakes*
How to protect MC in your programs

HARDWARE

40 **4K/8K ROMs in One ZX80** ..... *Rubesch*
How to use both ROMs

41 **Getting Loaded** ..... *Helton*
An external LED monitor

REVIEWS

42 **Machine Language Programming Made Simple** ..... *Wren-Hilton*
Book review

44 **The Exploding Bookshelf** ..... *Deeson*
ZX80/81 books: An overview and some details

GAMES

46 **6 Shooter** ..... *Dighera*
Win the kewpie doll

47 **Isolation** ..... *Nisbet*
Trap your opponent

Staff

Publisher/Editor-in-Chief	**David H. Ahl**
Managing Editor	**Paul Grosjean**
Contributing Editor	**David Ornstein**
Secretary	**Elizabeth Magin**
Art Director	**Susan Gendzwil**
Assistant Art Director	**Diana Negri**
Typesetters	**Jean Ann Vokoun**
	Maureen Welsh
Financial Coordinator	**William L. Baumann**
Personnel and Finance	**Patricia Kennelly**
Customer Service	**Ralph Loveys**
Circulation	**Frances Miskovich**
	Carol Vita
Advertising Sales Manager	**Jim Beloff**

MEMBER MTA

Index to Advertisers

Andover Software	30
Books for the ZX81	Cover 3
Byte-Back	5
Colossal Computer Cartoon Book	17
Compusoft	12
Computer Coin Games	17
Computers and Math	39
Emvee	30
Ezra Group	33
Future of Personal Computers	17
Gladstone Electronics	10-11
Intellectual Games	12
J.K. Greye	23
JRS Software	23
Lamo Lem	20
Leading Edge	Cover 4
L. J. H. Enterprises	23
Memotech	2
Mindware	13
Quicksilva	43
Reiter	36
RKL Systems	33
Sinclair Research	24-25
Softsync	3
Synchro-sette	7
SYNC Subscriptions	Cover 2
Time Data	36
Zeta Software	23

Volume 2, Number 2

SYNC (USPS: 585-490; ISSN: 0279-5701) is published bi-monthly for $16 per year by Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at New York, NY 10001, and at additional mailing offices.

Subscription rates: USA: 6 issues $16; 12 issues $30; 18 issues $42. Canada: $3 per year additional. Other foreign: $5 per year additional. U.K. Air: 6 issues £13. Minimum charge card order $10.00. For information concerning advertising in *SYNC*, contact Jim Beloff, Ziff-Davis Publishing Company, One Park Avenue, New York, NY 10016 (Phone: 212-725-3485).

U.K. address: *SYNC*, 27 Andrew Close, Stoke Golding, Nuneaton CV13 6EL.

Postmaster: Send address changed to SYNC, P.O. Box 789-M, Morristown, NJ 07960.

A creative computing PUBLICATION

Lack of ZX81 memory giving you headaches..?

The Memotech 64K Memopak

The growth of interest in computer use caused by the introduction of the Sinclair ZX81 has made new and exciting demands on the ingenuity of electronic engineers. At Memotech we have focused our attention on the design of an inexpensive, reliable memory extension.

The Memopak is a 64K RAM pack which extends the memory of the ZX81 by a further 56K. Following the success of our 48K memory board the new memory extension is designed to be within the price range expected by Sinclair users. It plugs directly into the back of the ZX81 and does not inhibit the use of the printer or other add-on boards. There is no need for an additional power supply or for leads.

The Memopak together with the ZX81 gives a full 64K, which is neither switched nor paged, and is directly addressable. The unit is user transparent and accepts such basic commands as 10 DIM A(9000)

0-8K ...Sinclair ROM

8-16K...This section of memory switches in or out in 4K blocks to leave space for memory mapping, holds its contents during cassette loads, allows communication between programmes, and can be used to run assembly language routines.

16-32K...This area can be used for basic programmes and assembly language routines.

32-64K...32K of RAM memory for basic variables and large arrays.

With the Memopak extension the ZX81 is transformed into a powerful computer, suitable for business, leisure and educational use, at a fraction of the cost of comparable systems.

DEALER ENQUIRIES WELCOME

MEMOTECH

Memotech
7550 West Yale Ave.
Suite 220
Denver, CO 80227
(Phone: 303-986-0016)

Please debit my MASTER CHARGE/VISA* account number:

*Please delete whichever does not apply

Signature

Date

Please rush me:

	Quantity	Price	Total
64K RAM, Assembled		$159.95	
(Special Introductory Offer)			

NAME

ADDRESS

SC1

Postage $4.00

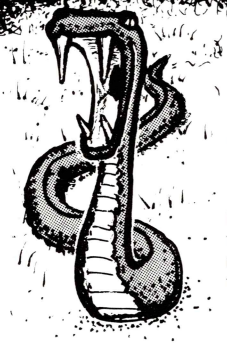
TOTAL

Memotech 7550 West Yale Avenue, Suite 220, Denver, Colorado 80227

Inspired by the adventure games originally written on large computers, these Adventures have been designed to run on your Sinclair ZX81 (and the ZX80 with 8K ROM).

The programs give you descriptions of where you are and what you see. It's up to you to tell the computer what to do. If you love a challenge, like to be baffled and enjoy jokes, you'll have hours of fun with these two Adventures.

The Adventures take you through rooms, caves and tunnels where you move and manipulate objects with short English commands.



ADVENTURE "A"

Your space ship is marooned on a strange planet but you can get out if you make the right combination of decisions. Written in machine language, this challenging adventure has over 100 words of vocabulary.

16K $19.95

ADVENTURE "B"

Enter the long lost Inca Temple, find your way through the tricky tunnels and corridors and you may find the lost treasure. Or you may be lost forever.

16K $19.95

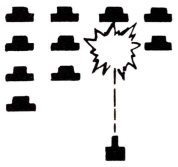
ALSO AVAILABLE

WRITE FOR FREE CATALOG

SUPER INVASION

"The best Sinclair game to hit the market." —SYNC Magazine. A moving graphics game with three levels of play. SUPER INVASION challenges your skill as you fire lasers at the attacking space invaders while maneuvering your space craft to avoid their deadly lasers.

1K $14.95



WALLBUSTERS

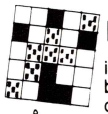
"A breakthrough in creating active display games."-SYNC Magazine. WALLBUSTERS challenges you to break through two barricades using nine balls and a curved bat. With seven levels of play, WALLBUSTERS is hard to beat. You'll be amazed at the superb graphics in this 1K game.

1K $14.95

ZXCHESS

This sophisticated chess game has seven levels of play and a detailed display of the board. You can change sides and even change levels of difficulty during a game. You can also start playing from any point in the game and if you get stuck, the computer will recommend a move.

16K $24.95



REVERSI

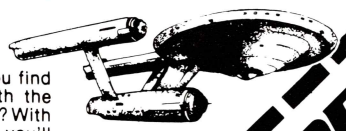
if you like Othello, you'll love REVERSI. With the board displayed, you can go first or let the computer go and you have a choice of starting positions.

1K $14.95

ROAD TO RICHES

What would you do if someone gave you a million dollars to invest? Would you make more money or lose it all? This investment game combines luck and strategy to challenge up to four players to wheel and deal their way to riches...or ruin.

16K $14.95



SPACE TREK

As commander of Starship Enterprise, you find yourself defending a galaxy overrun with the dreaded DRAKONS. Can you destroy them? With five levels of play and excellent graphics, you'll find SPACE TREK entertaining and challenging. Can only be used with the ZX81.

16K $14.95

SOFTSYNC, INC.

P.O. Box 480, Murray Hill Station, New York, N.Y. 10156

☐ Please send me ____ copies of ____ @ ____ each.

☐ Please send me ____ copies of ____ @ ____ each.

Foreign orders must be paid by International Money Order or foreign draft in U.S. Dollars drawn on a New York bank.

Please add $1.50 shipping & handling

New York residents add sales tax.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

letters

Help Wanted

Dear Editor:

As a rank amateur when it comes to computers, I wish to extend my thanks to *SYNC* for without it, I would probably just sit and stare at my ZX80 (8K ROM).

However, when it comes to programming, I am left out in left field. I have received some help from Radio Shack's *Computer Programming in Basic for Everyone* but some of the commands will not work, and I do not know how to convert them.

I would like to suggest that *SYNC* should feature an elementary course in programming for us duffers so that we may more fully enjoy our ZX80s.

Clifford Rose
180 Cabrini Blvd.
New York, NY 10033

Ed. — Your desire for programming help is shared by many of our readers. We are always looking for articles to help programmers at a variety of levels and for articles to help translate other Basics to the Sinclair version.

Dear Editor:

I have a ZX80 with an 8K ROM and 16K RAM. I would like to interface this with an Exatron stringy/floppy mass storage system. Do you know anyone who has done this and could help me? Do any manufacturers provide a card that would plug between the 16K RAM so that one could access the floppy files directly from software?

Jonathan A. White
PO Box 274
232 Hartford Ave.
Wilder, VT 05088

Ed. — We have not received any information on either point. If any reader can help, please drop a line to **SYNC.**

SGN and Inventory

Dear Editor:

David Ornstein is being a bit heavy handed in his SGN subroutine. When a number is divided by itself the answer is 1 and when a number is divided by its ABSolute value the answer is +1 or -1 according to the sign of the number. The only exception to this is zero. So all that is required in Listing 5 is

```
500 IF N=0 THEN LET SIGN=0
510 IF NOT N=0 THEN LET SIGN=N/ABS(N)
```

Next, I have five points on Dr. Justham's "Inventory."

1) I cannot see any need for lines 370-375 and similar. It seems to me that as line 310 puts N equal to the number of items then line 332 could read

```
FOR B=1 TO N
```

Lines 370-375 would then be unnecessary.

2) Line 3044 need not set up such a large array. DIM M$(1,J) is quite enough. Then in lines 3052 and 3068 use M$(1).

3) The possible error report confusion handled by the prompt in line 400 can better be handled with PEEK 16442. This returns the number of lines left on the screen including the two used for input only. So one could use:

```
435 IF PEEK 16442=4 THEN GOSUB 4000
4000 PRINT "PRESS NEWLINE TO CONTINUE"
4005 INPUT A$
4010 CLS
4015 RETURN
```

4) The note on line 305 says that the extra I$ and Q$ are needed for the "Delete" routine to run properly. But that routine could be amended by deleting lines 1800 and 1810 and continuing:

```
1820 FOR B=Y TO N-1
1830 LET I$(B)=I$(B+1)
1835 LET Q(B)=Q(B+1)
1850 NEXT B
1855 LET I$(N)=""
1860 LET Q(N)=0
1900 LET N=N-1
    etc.
```

5) Since the number 150 is used a number of times, memory savings can be made by adding

```
8 LET T=150
```

and then use T wherever 150 is used. This will save bytes whenever a given number is used four or more times. It has the added advantage that, if you want to change the 150 to some other number, you do not have to change all the lines.

Dr. Justham says that the program will handle up to 150 items so I assume that the 16K RAM is pretty well filled. If the suggestions above are used, a good bit of memory can be recovered, and thus the list can be extended.

Theodore F. Tott
Dadson and Butler
92B High Street
Gosport, Hampshire, PO12 1DS
United Kingdom

Make your "LITTLE" ZX81 *work like a* BIG *computer with* BYTE-BACK *modules*

16K MEMORY MODULE (M-16)

only **$69.95** NEW in stock

BYTE-BACK'S 16K memory module plugs right into the back of your ZX81 (or ZX80, with or without 8K Basic).

But unlike other 16K memory modules, up to three **BYTE-BACK M-16** memory modules can be connected at one time to get a total of 48K.

INSTANT INFORMATION WITH BYTE-BACK'S MD-1

In Stock! MODEM only $99.95

Use your phone to connect your "LITTLE" ZX81 or ZX80 to the "LARGEST" computer networks in the world. With **BYTE-BACK'S MD-1 MODEM** connected all you do is dial a phone number (usually local), press a few keys and watch the data appear on your TV screen.

RS-232 PORT INCLUDED

As an extra bonus an RS-232 port is provided to allow you to drive all standard RS-232 peripherals.

HIGH QUALITY BACK-BYTE PRINTER AVAILABLE SOON

48K MEMORY MODULE (M-48)

only **$179.95 NEW**

Wired and Tested: $189.95

BYTE-BACK'S BB-1 CONTROL MODULE

$59.00 In Stock!

- **8 Independent Relays** *(with LED status indicators)*
- **8 Independent TTL Inputs** *(with Schmitt trigger buffers)*

ALL MODULES CARRY 90 DAY WARRANTY

REMEMBER: With* BYTE BACK *modules you are NOT limited to using only one module at a time!

☐ M-48 Kit	$179.95	£90	☐ M-16 Kit	$69.95	£35
☐ M-48 Wired and Tested	$189.95	£95	☐ M-16 Wired and Tested	$79.95	£39
☐ BB-1 Kit and Manual	$59	£30	☐ M-16 Blank PC Board	$19.95	£10
☐ BB-1 Wired and Tested and Manual	$69	£35	☐ M-48 Blank PC Board	$19.95	£10
☐ BB-1 Blank PC Board and Manual	$29	£15	☐ Modem Wired and Tested	$119.95	£59
☐ Modem Kit	$99.95	£49	☐ RS-232 Port (W & T); not shown	$59.95	£30

Shipping and Handling $4.95

ORDER PHONE (803) 532-5812

Bill My ☐ VISA ☐ Mastercharge

Exp. Date _____ Card No. _____

Name _____

Address _____

City/State/Zip _____

Mail to:

BYTE-BACK Co. • Rt. 3, Box 147 • Brodie Rd. • Leesville, S.C. 29070

- VISA
- MASTERCHARGE
- CHECKS

ORDERS MAILED
First Class (U.S.A.)
Air Mail (England)

sync notes

Martin Wren-Hilton

SYNC Notes: U.K.

The Second ZX Microfair Report

Following the enormous success of the first ZX Microfair, the second ZX Microfair was held in Central Hall, Westminster, London, England, on January 30, 1982. Twice as much floor space and twice as many exhibitors ensured that everyone was catered for, that is, everyone who endured the two to three hour wait to get in. The queues outside doubled round the building as thousands waited for the doors to open at 10:30 a.m. Meanwhile, inside, the seventy-one exhibitors were making final adjustments to their stands.

Once inside, the ZX enthusiast was confronted by row after row of ZX80 and ZX81 suppliers with hundreds of products on show. These varied from simple 'addition' programs to disk drives with network controllers. The 'in' things at the show were high-speed machine code games, a wide variety of books to suit all requirements, and hardware accessories.

Some of the more spectacular demonstrations included a prototype color board by Haven Hardware, a 5 1/4 inch floppy disk drive by Macronics, a hi-res graphics controller and speech board by Quicksilva and a joystick controller by Microgen. Machine code games were all the rage—*Defender, Asteroids,* and *Invaders* by Quicksilva; *Centipede* by dK'tronics; *Galaxians* by Artic Computing, to name but a few. Many ZX users have complained about the miniature keyboard on the Sinclair computers, and, as a result, some twelve firms were demonstrating and selling full-size keyboards. However, even with a full size keyboard, game playing can lead to finger ache. Microgen's joystick controllers allow up to two people to play complicated games with ease.

Sinclair was there selling ZX81s, RAM packs, and ZX Printers, and answering questions related to the ZX80 and ZX81. Another group answering queries relating to these computers was the National ZX80/81 Users' Club founded by Tim Hartnell. Hints and tips were exchanged and a great deal was gained from this user-to-user conversation. *SYNC* was represented by Hazel Gordon, Creative Computing's U.K. representative, and myself. A number of people took out subscriptions on the spot. Some of the smaller local clubs were running stands and selling newsletters.

If a prize were to be given for the most effective and stunning use of the graphics on an unmodified ZX81, it would surely have to go to J. K. Greye Software for their *3D Monster Maze* which actually shows a moving, walking Tyrannosaurus Rex wandering around an apparently three-dimensional lair. It has to be seen to be believed.

If ZX80/81 related books were more your scene, then you had some 47 titles to choose from. If, on the other hand, you wanted your ZX80/81 to control your house or apartment then the RD8100 system would have attracted your attention. It comprises a large console which plugs into the back of the ZX81 with option Logic Input/Output, Analogue Input, Analogue Multiplexer/Amplifier, Analogue Output, Real Time Clock, and Light Pen System. Many other firms were selling Logic Input/Output units at very competitive prices.

Two disk drive systems were on show—one prototype by Monolith and one fully operational 5 1/4 inch unit by Macronics. The Macronics drive, called F.I.Z. (Floppy Interface for ZX81), comes with 2K operating firmware in ROM and 2K disk work RAM. The drive itself, a Shugart SA400, comes in an attractive cabinet which also houses the power supply. The interface is TTL which eliminates the need for expensive disk controllers. The F.I.Z. managed to load an 8K program in just 22 seconds. The capacity of each disk is 43250 bytes organized as ten 128 byte records per each of the 34 tracks.

The industrial applications of the ZX81 were highlighted by Mindware whose Barscan system allows the ZX81 to read optical bar codes like those found on the side of most merchandise. With some suitable software, it would be possible to load programs using this system. Businesses and small firms were catered for by Hilderbay's *Payroll* and very fast *Stock Control* programs. To go with these and other more complicated programs is the Memotech low-cost 64K RAM pack announced at the show. This pack allows 16K Basic programs with 32 K of variables and a further 16K of paged memory. Another 64K RAM pack, called the SUPER Z, was available from Audio Computers.

"Big Ears" is a voice recognition unit for the ZX81 from William Stuart Systems. This unit consists of a microphone, preamplifier, analogue frequency filters, and digital interface and comes complete with software.

The show finished at 8:30 p.m. and a reception was held for exhibitors at the "Westminster Arms," a local pub, by Mindware Co., where the first annual Rosetta Stone Award was made for the outstanding independently developed product for the ZX80/81 computers. The award was won by Dr. Ian Logan for his work in decoding the 8K ROM.

SYNC Program Listings

Readers should note the following conventions used in the program listings in this issue:

or • = Used in PRINT statements to show necessary spaces.

"A" (shift) = Used in PRINT statements to indicate graphics; in this case use the graphic on shift A.

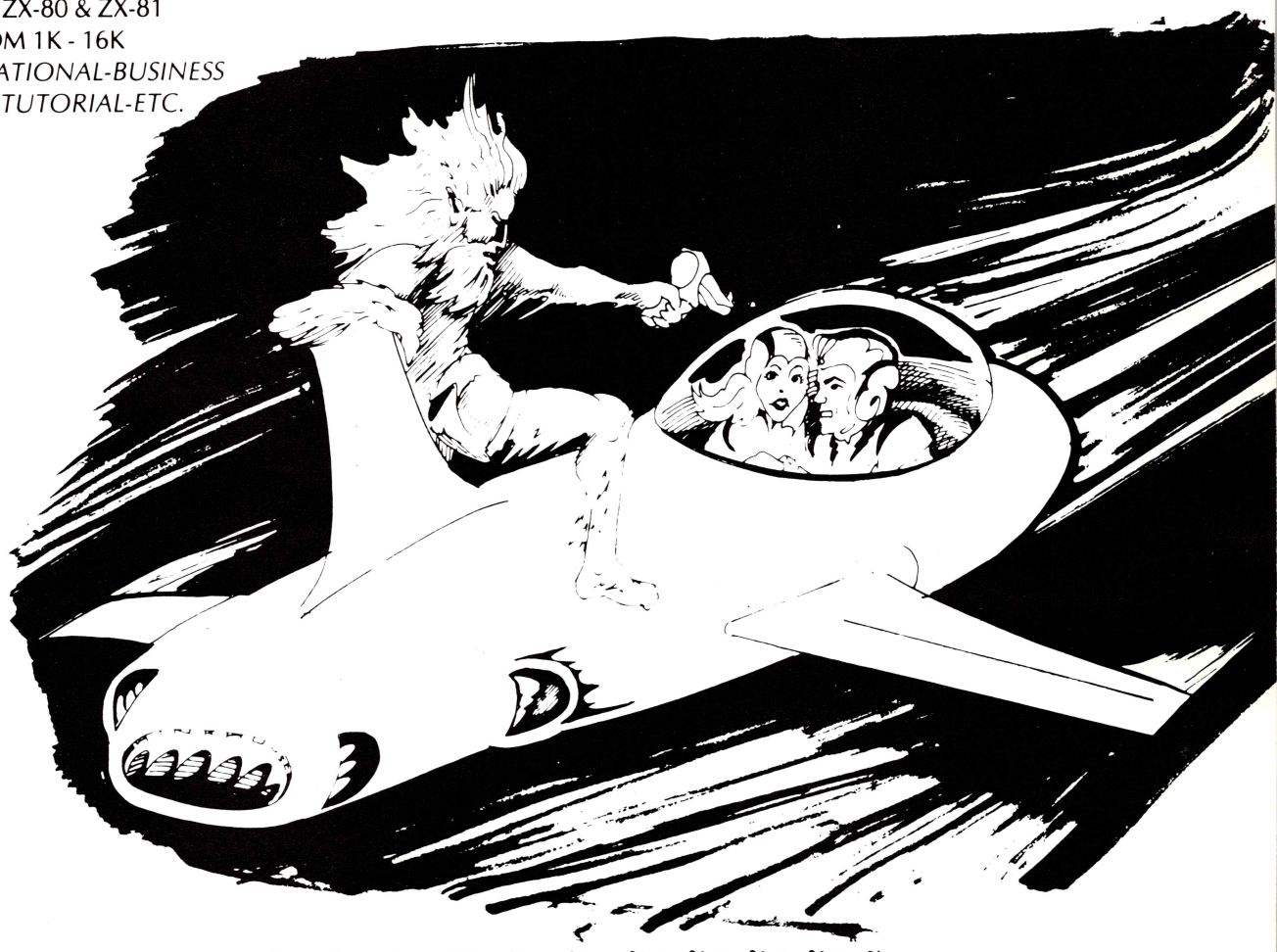
INPUT = Used in PRINT statements to show that the keyboard key or token should be used instead of spelling out the word.

Martin Wren-Hilton, U.K. Correspondent to SYNC, 4 Little Poulton Lane, Poultonle-Fylde, Backpool, FY6 7ET, United Kingdom.

SYNCHRO · SETTE

THE SUBSCRIPTION MAGAZINE FOR THE ZX-81 MICROCOMPUTER

FOR THE ZX-80 & ZX-81
8K ROM 1K - 16K
GAMES-EDUCATIONAL-BUSINESS
GRAPHICS-TUTORIAL-ETC.



12 Mo. Issues — 6 Bi-Month
CASSETTES
AT LEAST 6 PROGRAMS EACH

$39.50

MasterCard VISA

Ill. Residents add $2.07 tax
outside USA add $10.00

Ask for operator #383

24 HR. HOT LINE 800-543-1300

IN OHIO - 800-582-1364

THE S & S COMPANY
388 W. Lake Street
Addison, IL 60101
(312) 628-8955

perceptions

David B. Ornstein

The ZX80/81 Video Display System

Introduction

In this issue I will discuss the ZX80/81's display system. As this is a very involved technical subject, let us review how a television picture is generated.

The main element in a television is the CRT (Cathode Ray Tube). A basic CRT is shown in Figure 1. The tube, made of glass, is pumped free of air, creating a vacuum. Electrons are generated at the rear of the tube and shot forward toward the phosphor-cover of the screen. When these electrons hit the phosphor, it turns white. Varying the number (i.e., the intensity) of electrons shot at the phosphor changes the level of grey obtained.

If no other elements were involved, you would simply get a spot at an arbitrary place on the screen. To move the beam of electrons to different parts of the screen, deflector yokes are used. These are coils of wire that create a magnetic field. Circuitry in the TV causes the beam to move across the screen horizontally. When it reaches the extreme right of the screen, a pulse is applied to the system's control circuitry. This pulse, called horizontal sync, causes the beam to move back to the left side of the screen. As it moves back, it also moves down one scan line. This action is shown in Figure 2. When the beam reaches the bottom right hand corner of the screen, another pulse is applied to the system's internal circuitry. This signal, known as vertical sync, causes the beam to travel back to the top of the screen. A horizontal sync pulse, applied at about the same time, causes it to travel back to the left. The scanning process then repeats. The time that the beam spends returning, either to the left side or to the top of the screen, is known as retrace time. During this time, no electrons are shot at the screen; thus the name: blanking.

In the United States, a frame is scanned about every 16 ms (milliseconds, thousandths of a second). Thus, the frame rate of 60 Hz is established; the TV receives vertical sync 60 times per second. Horizontal sync is established at approximately 15 KHz (i.e., 15 thousand pulses per second).

Figure 1.

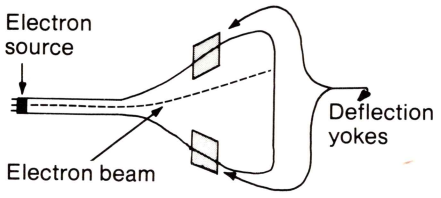
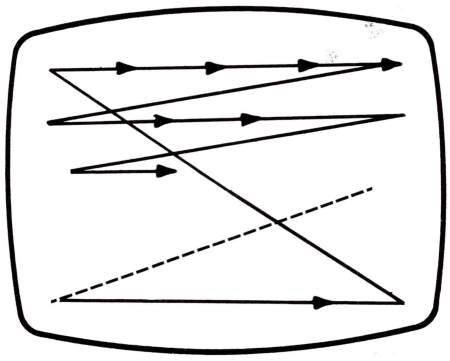


Figure 2.



The Display

The ZX80/81 video display is created almost entirely by software. All sync (vertical and horizontal) is generated by I/O instructions in the display routine. When the system decides that it wants to display a frame on the screen, it follows these simple steps:

1) Load the HL register pair with the address of the display file.

2) Set bit 7 in H to a 1.

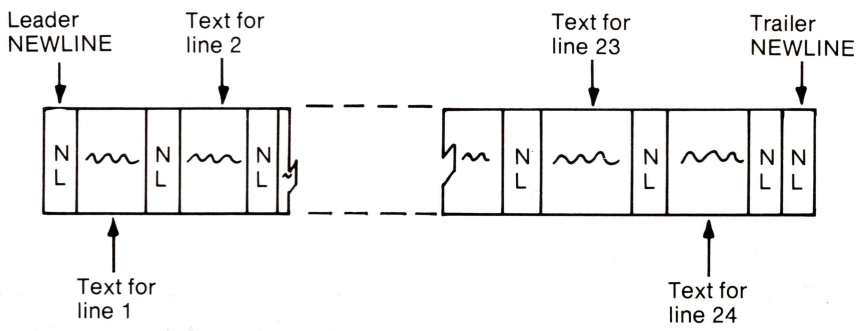
3) Load the R register with a time-out constant.

4) Jump to the address supplied in the HL register pair.

The designer of the ZX80/81 display system decided to use the Z80 microprocessor not only as a processing element to execute programs, but also as a fully software-programmable piece of logic.

The organization of the display file, shown below in Figure 3, was very carefully chosen. When the system 'jumps' to the display file, the PC (Program Counter) is used to

Figure 3.



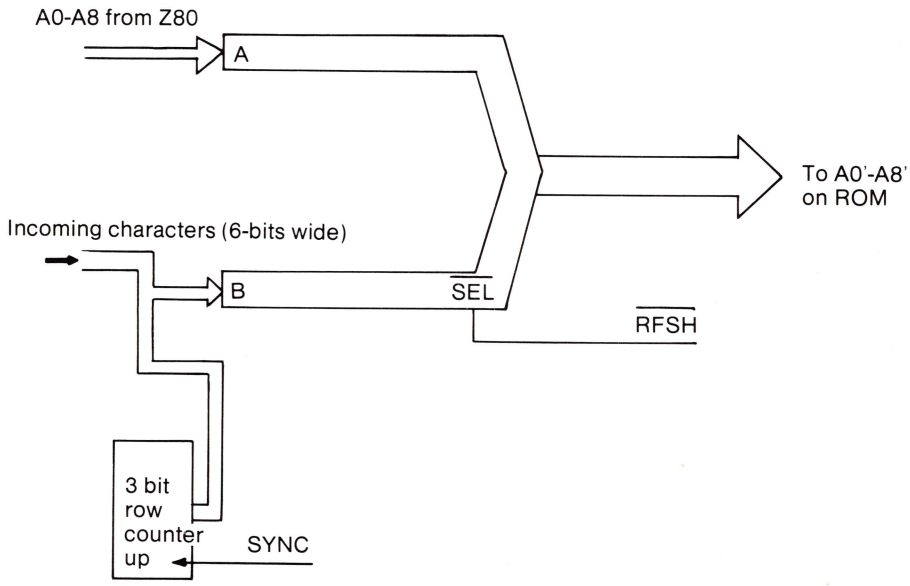
step through the sequence of characters to be sent out as video. The R (Refresh) register is used to keep track of the timing for the display. The Z80 processor automatically decrements it after fetching each instruction. Address line 6 is connected to the $\overline{INT}$ (Maskable-Interrupt) line. This means that whenever the A6 goes low (i.e., d6 of R=0) and Interrupts are enabled, the processor is interrupted from executing its present task. Interrupts are enabled only during the time that the system is displaying, so as to disallow spurious interrupts.

Let us follow through the display of a frame on the screen. The system first performs a read of the keyboard port (port FE hex, i.e., A0=low). This causes $\overline{RD}$, $\overline{IORQ}$, and A0 to become low. This, in turn, causes a set/reset type flip-flop to be set. The output of this flip-flop, sent through a ripple-through arrangement of preset/clear D-type flip-flops, becomes the SYNC signal. This SYNC flip-flop is reset by a write to *any* I/O port. It is also pulsed by *any* I/O request ($\overline{IORQ}$=0). The display routine is also indirectly responsible for the keyboard scanning. After performing calculations on the data obtained through the initial IN instruction, it writes to an I/O port, ending Frame Sync. Frame Sync lasts for approximately 388μs (1262 T-States; a T-State's a system clock cycle, and it lasts for 307ns). The system then executes the jump to the display file. In the ULA (Undedicated Logic Array) on the ZX81, the fact that the processor is executing instructions at an area in "memory" whose address has A15 set (i.e., high) is detected. This important control signal is designated $\overline{DISP}$.

Reviewing the system's schematic, you will notice that the data bus is "split" into two distinctly separate buses with eight resistors. These buses are specified as D0-D7 and D0'-D7'. The D0-D7 side of the resistors is connected directly to the Z80's data bus pins.

When the $\overline{DISP}$ signal is active and $\overline{MI}$ (Instruction Fetch) is active and D6' is low, a set of eight open collector inverters drives a 00 hex onto the D0-D7 bus. This is the machine code for the Z80's NOP (No OPeration) instruction. As the processor steps through the display file, it receives NOPs and executes them.

Figure 4.



Meanwhile, the actual characters in the display file are sent out from RAM (Random Access Memory) onto the D0'-D7' bus. They travel through a buffer hatch and into one side of a set of multiplexors. The general configuration of the MUXs is shown in Figure 4. (A multiplexor is a handy piece of logic used to select, as output, one of two inputs. A select line, designated $\overline{SEL}$, is used to choose between the two inputs.)

During the execution phase of each instruction, the Z80 performs a refresh: the $\overline{RFSH}$ line becomes active, the R register goes out on A0-A7, and the I register goes out on A8-A15. The ZX80/81 ROM holds the character generator (map of the characters). Each character is set up as an 8x8 matrix.

As each refresh cycle executes, the character code, combined with the current row address (0 . . . 7) from the three-bit row counter (incremented by Horizontal SYNC), goes to the ROM as the lower nine bits of its address. The rest of the address for the ROM comes directly from the Z80 (i.e., the I register, as this is a $\overline{RFSH}$ cycle). Thus the I register serves to hold the base address for the character generator which resides in the last 512 bytes of the ROM (4K or 8K).

The eight bits of pixel data come out of the ROM and travel along the D0'-D7' data bus and into a shift register. The output of this shift-register, clocked at 6.5 MHz, combined with the SYNC pulse, is video.

Earlier it was noted that D6' was required to be low for a NOP to be forced onto the bus. The code for the NEWLINE line-delimiter in the display file was very carefully chosen to be 76 hex (bit 6=1); 76 hex is a HALT instruction on the Z80. When the system gets to the end of displaying a line of characters, it encounters a NEWLINE character. This instruction is allowed to pass onto the D0-D7 bus and the processor executes the HALT instruction.

All this time, with every instruction cycle, the R register has been decrementing. Its setup value is carefully chosen so as to interrupt the process at the end of each pixel line. The interrupt handler deals with several things. First, the Z80 acknowledges the interrupt with a special "interrupt acknowledge" cycle. This is signified by both $\overline{MI}$ and $\overline{IORQ}$ becoming active (i.e., low). The SYNC flip-flops notice this and cause a SYNC pulse to be sent out to the video-sync combination/modulator circuitry. This pulse is horizontal sync and it lasts for 6.15μs (20 T-States). Next, the

ZX81 SOFTWARE On Cassette

GAMES PACKS

1 for 1K ZX81 & 8K ROM ZX80. Eight fantastic programs for the unexpanded ZX81, including DIGICLOCK, 9-LIVES, REACTION TEST, GOBBLER and PATTERNS.
$9.95 ($12.95 in Canada)

2 for 16K ZX81. Four programs written in BASIC for the expanded ZX81. PONTOON, FRUIT MACHINE, OXO, and BIO— RHYTHMS.
$9.95 ($12.95 in Canada)

3 for 16K ZX81 and 8K ROM ZX80. Two programs for expanded ZX81 to keep you entertained for hours! 3-D OXO is written in machine code and is hard to beat. MARS RESCUE is a compulsive adventure game.
$9.95 ($12.95 in Canada)

4 for 16K ZX81. ZOMBIES — escape as they chase you around Zombie Island. Lure them into the pits, but don't fall in yourself. MOUNT MAYHEM — can you reach the 20,000 foot summit? Look out for Yetis and other hazards!
$9.95 ($12.95 in Canada)

MULTIFILE

Data Storage System
An amazingly versatile multi-purpose filing system for the 16K ZX81. The program is menu-driven, and number, size and headings of files are user-definable. Both string and numerical files are catered for. Files may be created, modified, replaced, and searched, and are protected by an ingenious foolproof security system. Output to the ZX printer is also provided. The program comes on cassette, together with three quality data cassettes for file storage, and comprehensive documentation, describing a host of applications for both business and personal use. Supplied in an attractive storage case. If your ZX81 is bored with playing games, then this program will give it plenty to think about!
$29.95 ($39.95 in Canada)

DICTATOR

Fantastic new adventure game for 16K (or greater) ZX81. You have just become 129th ruler of Ritimba with a single goal in mind: take full advantage of the situation for your own good. You have to deal with a handful of factions: unruly army, downtrodden peasants — but you have the secret police on your side.
$14.95 ($19.95 in Canada)

CONSTELLATION

Turn your ZX81 into a telescope with this amazing 16K program. Produces a simulation of the night sky as seen from any position on Earth at any chosen time this century. You may point your telescope in any direction, move it up, down, left or right, zoom in or out. Stars may be displayed by magnitude or constellation
$14.95 ($19.95 in Canada)

ZX CHESS

(for ZX81 and 8K/ZX80 both with 16K RAM)

A challenging chess programme, written in machine language, designed to operate in the ZX81 fast mode. ZX Chess allows you to select from 6 levels of play, choose either black or white, and enables castling and en passant moves. Unique "self-running" feature: you start the tape and when the chess board appears on the screen, start your game.
ZX CHESS! Melbourne House. **$24.95** ($29.95 in Canada)

STAR TREK

The classic computer game in which you trek across the galaxy in search of Klingons to zap with your phasers and photon torpedoes. You have long and short range scanners to help you find them, Starbases to refuel your ship at and, of course, various witty comments from the crew. 16K.
$9.95 ($12.95 in Canada)

BOOKS

Not Only 30 Programs for the Sinclair ZX81: 1K.
Not only over 30 programs, from arcade games to the final challenging Draughts playing program, which all fit into the unexpanded 1K Sinclair ZX81 but also notes on how these programs were written and special tips! Great value!
$14.95 ($16.95 in Canada)

Machine Language Programming Made Simple for the Sinclair
A complete beginner's guide to the computer's own language—Z80 machine language. Machine language programs enable you to save on memory and typically give you programs than run 10-30 times faster than BASIC programs.
$19.95 ($23.50 in Canada)

Understanding Your ZX81 ROM
A brilliant guide for more experienced programmers by Dr. Ian Logan, this book illustrates the Sinclair's own operating system and how you can use it. Includes special section on how to use machine code routines in your BASIC programs.
$19.95 ($23.95 in Canada)

ZX81 ROM Disassembly
Part A. Covers ROM locations from 000H to OF54H and includes all functions except for routines used in floating point calculator. A must for the experienced programmer.
$14.95 ($16.95 in Canada)

VU-CALC

VU-CALC. Constructs, generates, and calculates large tables for analysis, budget sheets and projections. Up to 26 columns of figures or data can be entered, plus user definable formulae capable of relating any one or more position in the table to any other defined position.
$29.95

VIEWTEXT

A ten page information display system for the 16K ZX81. Can display both text and graphics in any sequence with variable speed. Many applications including shop window displays, educations, animation, etc.
$14.95 ($19.95 in Canada)

ZXAS

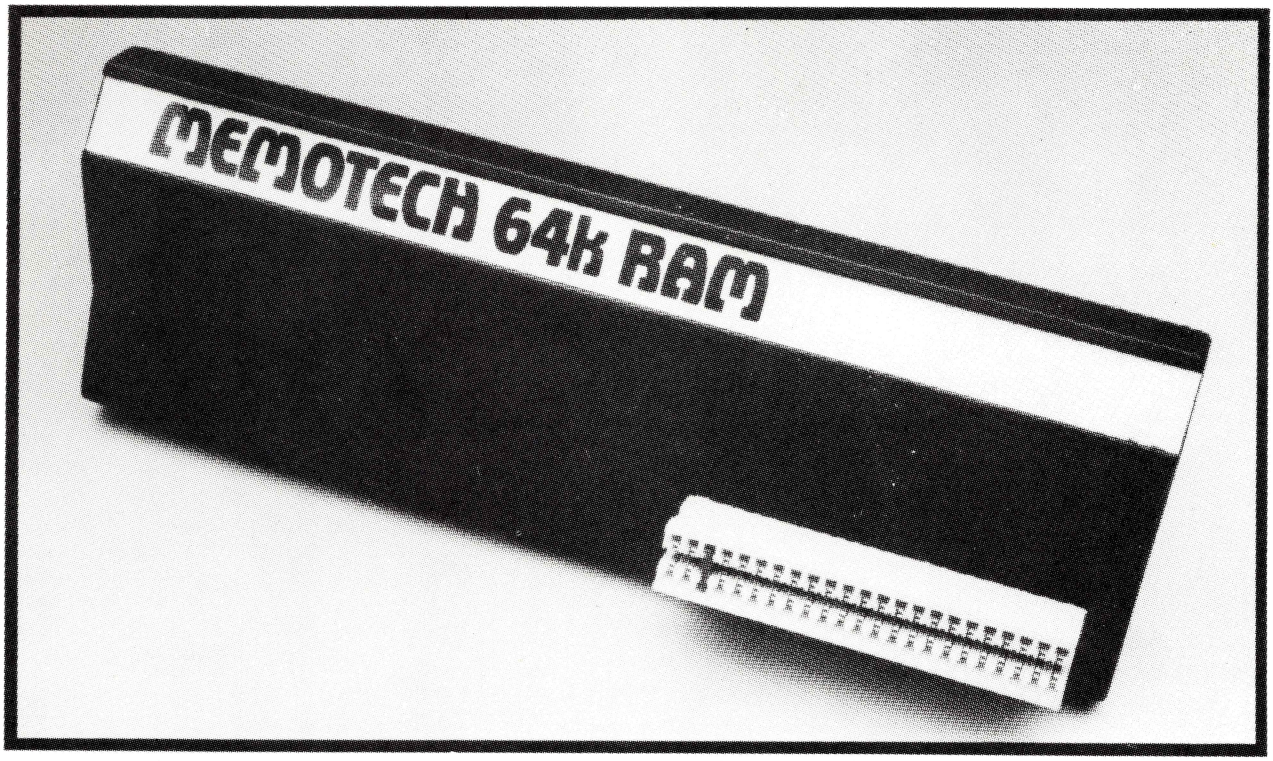
Now you can use the full power of the Z80 microprocessor without having to laboriously POKE in instruction codes. This full specification Z80 assembler assembles all the standard Zilog mnemonics, which are simply written into REM statements (more than one per line is allowed) within your BASIC program. When assembled, the assembly listings, together with assembled codes and adresses, are displayed on the screen. The assembled code is executed by USR. The program occupies 5K, is situated at the top of the memory, and is protected from overwriting. This means that ZXAS may be used in conjunction with ZXDB (see below), providing an extremely powerful machine code system normally only found on very expensive computers.
The program is available for both the ZX81 and the 8K ROM ZX80, and in both cases, the 16K RAM pack is required. Despite the low price, ZXAS is a FULL-SPECIFICATION assembler, and is a must for all serious ZX users. Full documentation on how to use the assembler (including a list of the mnemonics) is supplied.
$9.95 ($12.95 in Canada)

ZXDB

The perfect complement to the ZXAS assembler, ZXDB is a complete combined machine code disassembler and debugging program. Like ZXAS, it is itself written in machine code for compactness, and may be used in conjunction with ZXAS, still leaving about 9K of memory for your own program.
Apart from the DIASSEMBLER, the program has features including SINGLE STEP, BLOCK SEARCH, TRANSFER AND FILL, HEX LOADER, REGISTER DISPLAY and more, all of which are executed by simple one key commands from the keyboard. All in all, an extremely powerful programming aid, well worth the money for the disassembler alone!
$9.95 ($12.95 in Canada)

GLADSTONE ELECTRONICS

64K-$159



The Memotech 64K Memopak

The growth of interest in computer use caused by the introduction of the Sinclair ZX81 has made new and exciting demands on the ingenuity of electronic engineers. At Memotech we have focused our attention on the design of an inexpensive, reliable memory extension.

The Memopak is a 64K RAM pack which extends the memory of the ZX81 by a further 56K. Following the success of our 48K memory board the new memory extension is designed to be within the price range expected by Sinclair users. It plugs directly into the back of the ZX81 and does not inhibit the use of the printer or other add-on boards. There is no need for an additional power supply or for leads.

The Memopak together with the ZX81 gives a full 64K, which is neither switched nor paged, and is directly addressable. The unit is user transparent and accepts such basic commands as: 10DIM A(9000)

0-8K. . .Sinclair ROM
8-12K. . .Memopak memory which can switch in or out in 4K blocks to leave space for memory mapping.
12-16K .Memopak memory which holds its contents during cassette loads and allows communication between programmes.
16-32K. . .This area can be used for basic programmes and assembly language routines.
32-64K. . .32K of RAM memory for basic variables and large arrays.

With the Memopak extension the ZX81 is transformed into a powerful computer, suitable for business, leisure and educational use, at a fraction of the cost of comparable systems. **($219. in Canada)**

GLADSTONE
Electronics 901 Fuhrmann Blvd., Buffalo, NY, 14203

Telephone Orders
(716) 849-0735

Have your VISA or MASTERCARD READY.
Shipping and handling charge $1.50 per order books and cassettes, $3.00 per order 64k Memopak. Sorry No C.O.D.s.
Memopak 64K RAM is guaranteed for 90 days. Tapes carry a replacement warranty.
In Canada: **Gladstone Electronics,** 1736 Avenue Rd., Toronto, Ont., M5M 3Y7 Telephone (416) 787-1448.

CHARGE TO:
☐ VISA ☐ MASTERCARD

account number:

Expiry date

Signature

Date

Please rush me:

	Quantity	Price	Total
64K RAM		159	

NAME

ADDRESS

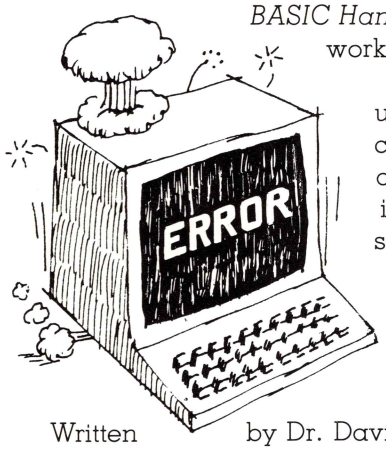
SHIPPING

TOTAL

To: Gladstone Electronics, 901 Furhmann Blvd., Buffalo, N.Y. 14203 (716) 849-0735

When your computer won't speak your language, you need a basic handbook.

As a matter of fact, everyone who works in BASIC needs *The BASIC Handbook*. It is *the* definitive reference work on the subject of BASIC.

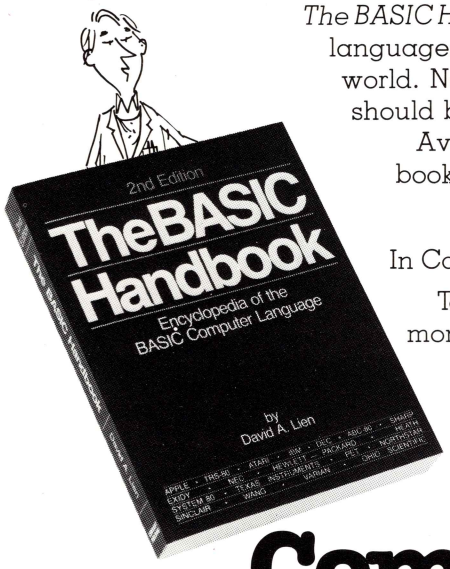


The BASIC Handbook is an easy-to-use encyclopedia of nearly 500 words covering the "dialects" used by virtually every BASIC-speaking computer in the world. But more than that, it's a simple, step-by-step guide to translating programs from one computer to another. So now you can actually *use* software printed in magazines and elsewhere, no matter what computer you own.

Written by Dr. David A. Lien, author of the *Tandy TRS-80 Level I User's Manual* and the *Learner's Manuals* for the Epson MX printers, this completely revised Second Edition contains almost twice as many entries as the best selling First Edition, making it by far the most up-to-date BASIC reference book you can buy.

The BASIC Handbook

Extensively indexed and cross-referenced, *The BASIC Handbook* gives you 480 pages packed with the information you need to be a better programmer. And if, after 30 days you don't agree it's indispensable, send it back. We'll return your money.



The BASIC Handbook is available in several languages and accepted throughout the world. No one who programs in BASIC should be without it.

Available at better computer and book stores, or call **(800) 854-6505**

In California **(714) 588-0996**

To order by mail, send check or money order for $19.95 (California residents add 6%), plus $1.65 shipping and handling. Overseas orders send $19.95 plus $2.38 surface shipping and handling.

VISA

MasterCard

CompuSoft® Publishing

1050-E Pioneer Way, Dept. E, El Cajon, CA 92020

"pixel lines left" counter (the C register) is decremented. If the counter becomes zero, then it is reset to 8 (the number of pixel lines per character), and a delay is executed to allow horizontal retrace to occur.

The "lines per frame" counter (the B register), initially 24, is then decremented. If it reaches zero, the frame is done and the display-routine performs a RETurn instruction, permitting the system to continue with the task it was performing before it began scanning the frame.

If either the "pixel lines left" counter or the "lines per frame" counter are non-zero at this point, the R register is reloaded with the proper value, interrupts are reenabled (as the Z80 automatically disabled them when the original interrupt occurred), and a jump is made back into the display file, at the beginning of the same line of characters, so as to scan the next pixel line of the screen.

On the ZX80, constant, "flicker-free" video is not available. This is because, as you can now see, the Z80 microprocessor must be explicitly executing the display loop. On the ZX81, constant video is generated with a bit of additional circuitry. This circuitry is essentially a 60 Hz clock, whose output is connected to the $\overline{\text{NMI}}$ (Non-Maskable Interrupt) line on the Z80. The handler (i.e., the subroutine called when the interrupt occurs) for this interrupt scans a single frame on the TV screen and RETurns, allowing the system to get back to the task at hand, namely, executing your program. When in SLOW mode, interrupts (NMI) are being generated, and your program is being executed only during vertical retrace time: thus the name. Personally, unless I am working with graphics, I prefer the speed.

A Note to Readers

I enjoy providing technical support services and information for *SYNC* readers through this column. Since I like to cover a broad range of material in order to reach all our readers in one way or another, please send any ideas for articles you would like to see written to me: c/o Heuristics, 25 Shute Path, Newton, MA 02159.

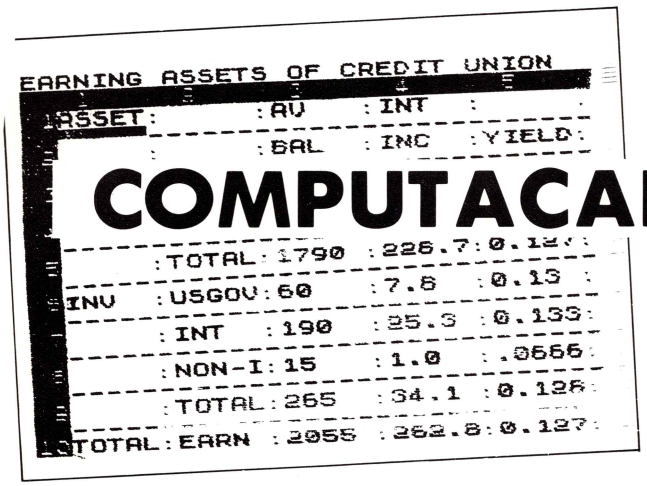
Until next time, same relativistic time period, same non-Euclidian universe. ■

OTHELLO FOR 1K

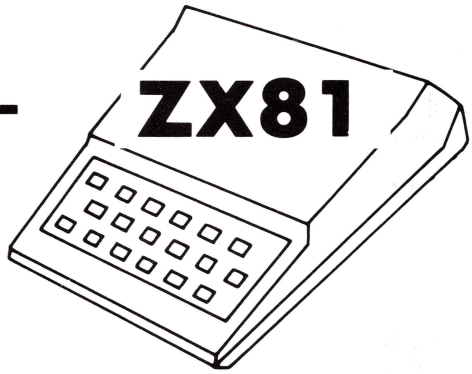
GRAPHICS, HIGH-SKILL PLAYING ALGORITHM

$7.50 (U.S.) $10.00 (NON-U.S.)

INTELLECTUAL GAMES
193 PEACEABLE ST.
RIDGEFIELD, CT 06877


EARNING ASSETS OF CREDIT UNION
ASSET: :AV :INT :
: :BAL :INC :YIELD:
: TOTAL: 1790 : 226.7: 0.12
INV : USGOV: 60 : 7.8 : 0.13
: INT : 190 : 25.3 : 0.133
: NON-I: 15 : 1.0 : .0666
: TOTAL: 265 : 34.1 : 0.128
TOTAL: EARN : 2055 : 262.8: 0.127

COMPUTACALC + ZX81



. . . a powerful combination at a bargain price!

MINDWARE introduces **COMPUTACALC,** a low priced software package for the **ZX81. COMPUTACALC** is an electronic worksheet with an almost unlimited range of applications.

CONSIDER:
Financial forecasts
Engineering calculations
Productivity analyses
Pricing
Job costing
Estimating
Production scheduling

A versatile tool for **$39.95**
Requires 16K RAMPAK

MINDWARE has a catalog of software, hardware and books for the **ZX81.**

Write
MINDWARE CO.
70 Boston Post Road
Wayland, MA 01778

MINDWARE CO.
Products that supplement nature's computer

8K ROM
1K RAM

Kitchen Sync

Alan Groupe, Michael Tardiff, and Ivan Zatkovich

Using Fun Features to Make a Great Toy

The first thing we did when we finished building our ZX81 kits was to play with them—poking through the manual, pressing new keys to see what they did, writing little test programs to see what could be done. It did not take us long to decide that the ZX81 had some features that made it a lot of fun to have around.

Of all the new features good old Clive included in the bigger, better 8K Basic, our favorites are SLOW mode, the PLOT command, and the INKEY$ function. In this column we will play with these features a bit, and end up showing you how to use these features to write a drawing program (similar to the famous mechanical Etch-A-Sketch) that was suggested by Paul Merchant, a sophomore in the Merrimack Valley High School in Penacook, New Hampshire.

Let us look at our three favorite new features one at a time.

The SLOW Mode

If you have had your ZX81 more than a week, you have no doubt discovered SLOW mode (called "compute and display mode" by Sinclair—the word "slow" has bad connotations in connection with computers). If you have not quite grasped what happens in slow mode, the following program will help:

```
10 PRINT "E";
20 GOTO 10
```

Now switch to fast mode, by pressing the SHIFT and F keys and then pressing ENTER. The screen will go blank and 0/0 will appear at the bottom. Press RUN, then ENTER. The screen will go dark for a short while and then come back with a screen full of E's.

Thrilling, no?

Press SHIFT D and ENTER. You are now in SLOW (sluggish, slothful, languorous—actually, we never saw much wrong with "SLOW" ...) mode. Press RUN and ENTER again. You get the same end result, but this time you see the E's zipping across the screen.

"That didn't take much longer than doing it in fast mode," you are thinking. You wonder why you would ever want to go back to FAST mode, right? Before you jump to conclusions, try the following program, first in FAST mode, then in SLOW mode:

```
10 FOR I=1 TO 15
20 PRINT I,SQR I
30 NEXT I
```

In FAST mode it should have taken about five seconds to print out the table of square roots, but in SLOW mode, over thirty. Obviously, when you are doing anything more than nothing at all, there is indeed a tradeoff in speed when you use SLOW mode, since your computer's mighty little Z80 microprocessor has to stop running your program every fraction of a second to make sure that the screen display is up-to-date.

It is important to realize that one of the most interesting features of FAST and SLOW is that they are *commands*, just like any of the others on the ZX81 keyboard, and, as such, they can be used within a program. In other words, you can execute most of a program in SLOW mode, but then go into "warp-drive" when you need to do heavy computations.

The PLOT Command

The next fun feature of the ZX81 is the PLOT command. In geometry-talk, PLOT plots x- and y-coordinates in the first quadrant. In real life, PLOT puts graphics "dots" on the screen. A "dot" is one quarter of a character-space. If you enter

```
PLOT 1,1
```

you get a dot in the lower left corner of the screen.

PLOT provides you with a very versatile tool for creating figures and drawings. You have the graphics set of the ZX81, which contains every possible combination of "quarters-of-a-character-space"(a.k.a. "dots") turned on or off, along with the ability to put (or "unput") a dot anywhere on the screen.

Run the following program in SLOW mode. (By the way, we have found that it is a lot easier to enter programs while in FAST mode, and use the screen flicker feedback to know when we have pressed a key. You will soon find that the slow mode can get awfully slow when you are entering a long program . . . try it and see.)

```
10 FOR I=1 TO 40
20 PLOT I,I
30 NEXT I
40 FOR I=1 TO 40
50 UNPLOT I,I
60 NEXT I
70 GOTO 10
```

If you do not get too excited by straight lines, try this recurring sine wave program:

```
10 FOR N=0 TO 63
20 PLOT N,10*SIN (4*N*PI/63)+2
0
30 NEXT N
40 FOR N=0 TO 63
50 UNPLOT N,10*SIN (4*N*PI/63)
+20
60 NEXT N
70 GOTO 10
```

The INKEY$ Function

Last comes the INKEY$ function which is probably the most useful command you have available for making your ZX81 respond easily to input. The value of INKEY$ is a character string that contains the key currently being pressed on the keyboard. If no key is being pressed, INKEY$ returns a null (empty) string.

To test out INKEY$, run the following program in SLOW mode:

```
10 PRINT INKEY$;
20 GOTO 10
```

The screen will blank out, and you will probably see a couple of characters in the upper left corner—do not worry about them. If you press a key, that key will be

displayed on the screen. But try pressing a key and holding it down. See lots of characters on the screen? INKEY$ keeps checking to see if a key is pressed, and keeps finding the one you are pressing.

By now, you have probably figured out that the few spurious ?'s in the upper left corner of the screen came from holding down the ENTER key too long after pressing RUN. If you did not get any question marks, you have a remarkably light and quick typing touch.

Putting SLOW, PLOT, and INKEY$ Together

Now that we have told you a little about our three favorite commands, let's tie them all together and write a program that will do something interesting. The following program will keep children, guests, and even computer haters busy for hours, doodling away—if you stop playing with it yourself long enough for them to get a chance.

```
10 LET X=31
20 LEY Y=21
30 PLOT X,Y
40 LET K$=INKEY$
50 IF K$="5" THEN LET X=X-1
60 IF K$="6" THEN LET Y=Y-1
70 IF K$="7" THEN LET Y=Y+1
80 IF K$="8" THEN LET X=X+1
90 IF K$="E" THEN CLS
200 GOTO 30
```

When you run the program in slow mode, a dot appears in the center of the screen. Press one of the arrow keys (unshifted). You will have a short line in the direction of the arrow you chose. Hold an arrow key down, and a longer line is drawn. Press E and the screen is erased, leaving the dot where you last left it. Go crazy.

If you can tear yourself away from this great toy (and it took us hours to tire), you might want to try a few improvements to the program. Remember, though, that the larger the program gets, the less memory there is for the screen display, and the slower the program runs. That is a fact of life not only with the ZX81, but with virtually every computer.

First improvement: get rid of the bug. What is it? You probably found out already. If you hold down the 8 key (right arrow), eventually the dot stops at the right edge of the screen and the program stops, leaving a "B/30" on the bottom of the screen. This means, "You tried to go off the screen." The same thing will happen with the 7 key (up arrow). But something even stranger happens if you go too far to the left, or too far down the screen.

Run the program again and bring the dot fairly close to the left of the screen (with the 5 key). Press "E" to erase the screen, then hold down the 5 key again. If you hold it long enough, the dot will reach the left side of the screen, then bounce back and begin moving to the *right* every time you hit the *left* arrow. If you move the dot down a couple of lines, you will find that now the dot moves *left* when you press the *right* arrow. Why does it go left when you want right, and vice-versa? Because every time you press the left arrow key, the value of x is decremented. X becomes a negative number. But the PLOT command uses the absolute value of the coordinates (the number, without the plus or minus sign), so -5 turns out to be 5, which moves the dot left, thus the mirror image effect.

You can do something about this, of course. If you add lines to the program to check whether the dot has reached an edge of the screen, you are doing what computer people call "bounds checking." If you try to move the dot off the edge of the screen ("out of bounds"), bounds checking notices, and tells the program not to let you do it.

Add the following lines to the program:

```
100 IF X<0 THEN LET X=0
110 IF X>63 THEN LET X=63
120 IF Y<0 THEN LET Y=0
130 IF Y>43 THEN LET Y=43
```

Now if you reach the edge of the screen, the dot stops, and further commands to move in that direction are ignored. And, if you think back to childhood (or to your children's childhood, if you were born before Etch-A-Sketch . . .), you will remember that the controls of an Etch-A-Sketch work just like this.

While you are thinking of that mechanical Etch-A-Sketch, did you ever wish that there was a way to move the pen without drawing a line, so that you could draw pictures or letters that did not touch each other? Well, here is one way a computer beats the mechanical model—it is easy to fix this program so that you can "lift the pen." Add these lines to your program:

```
 5 LET PEN=1
30 IF PEN=1 THEN PLOT X,Y
35 IF PEN=2 THEN UNPLOT X,Y
92 IF K$="D" THEN LET PEN=1
94 IF K$="U" THEN LET PEN=2
96 IF K$="M" THEN LET PEN=0
```

With these changes, you can move the dot drawing a line, move it erasing a line, and move it without changing the screen. All we have done is to create a new variable (we called it PEN) that keeps track of the state of the "pen" (which is what we will call the dot from here on). Every time the pen gets to a new location, the program checks the PEN variable to see whether to draw a dot, erase one, or do nothing. If you press the D key, PEN will be set equal to 1, which means "draw a dot." If you press the U key, PEN will be set to 2, which means "erase (undraw) a dot." And the M key sets PEN to 0, which means "move, but do not do anything," so pen movements are invisible on the screen.

As soon as you have tried moving the pen in "M" mode, you will want another feature—some way of figuring out where the pen is without drawing any lines. The following two program lines make a blinking dot appear at the current location of the pen whenever you press the F key:

```
133 IF K$="F" THEN PLOT X,Y
135 IF K$="F" THEN UNPLOT X,Y
```

This works in both "Draw" and "Erase" modes, but always turns the current position off when in "Move" mode. Why? We will leave you to figure that one out. If you want to always see where the pen is, you might want to remove the conditional parts of lines 133 and 135

IF K$="F" THEN

and make the current pen position always blink.

What else can you add? Keys to draw diagonal lines might be nice, and you can probably think of a lot more. But, as you have probably noticed by now, every new feature added leaves less room on the screen and makes the program run slower. If you have lots of memory, you will not run out of screen space, and only the decrease in speed will be a factor.

The Improved Sketch Program

If you have just 1K of memory, you might want to try this improved version of the program that uses less memory:

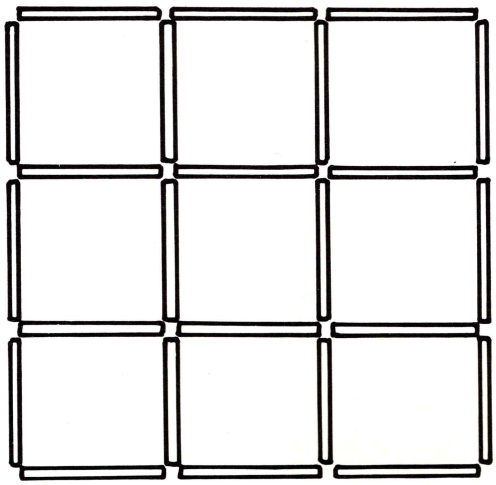
```
 10 LET PL=0
 20 LET X=31
 30 LET Y=21
 40 LET K$=INKEY$
 50 IF K$="5" AND X>0 THEN LET
X=X-1
 60 IF K$="6" AND Y>0 THEN LET
Y=Y-1
 70 IF K$="7" AND Y<43 THEN LET
 Y=Y+1
 80 IF K$="8" AND X<63 THEN LET
X=X+1
 90 IF K$="E" THEN CLS
100 IF K$="P" THEN PL=1-PL
200 IF PL=1 THEN GOTO 250
210 PLOT X,Y
220 UNPLOT X,Y
230 GOTO 40
250 PLOT X,Y
260 PLOT X,Y
270 GOTO 40
```

This version has a blinking cursor to mark the current position of the pen, and also lets you see the pen location when you cross over a black line. Pressing the P key turns the "write" function of the pen on or off. You can turn the pen off for going from one part of the screen to another without leaving a line, or use the turned-off pen to erase lines. ■

puzzles & problems

A Square Problem

We start off this puzzle session with a matchless problem from Mr. Ian Lowry of London, England. Mr. Lowry instructs us to arrange 24 matches into 9 squares as shown in the illustration at the right. Our problem is to remove 4 matches leaving 6 squares on the table. Thanks for sharing this puzzle with us, Ian. Merlin is sending you a copy of *Merlin's Puzzler 2* for your efforts. Nice going!



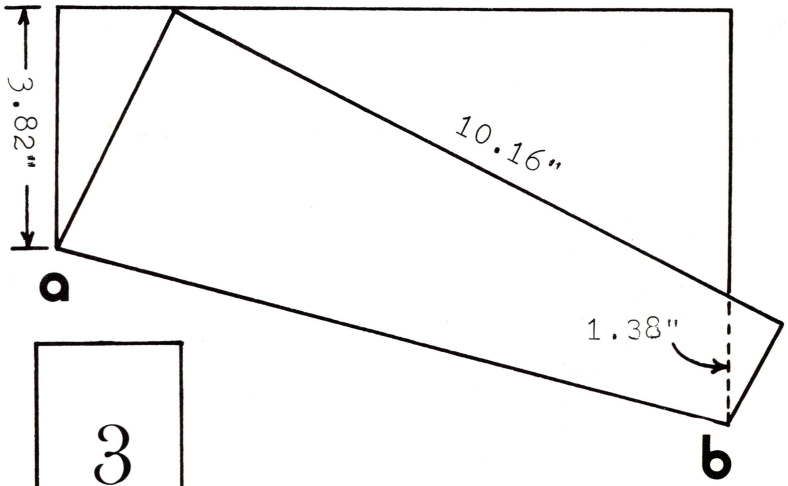
Another Square Problem

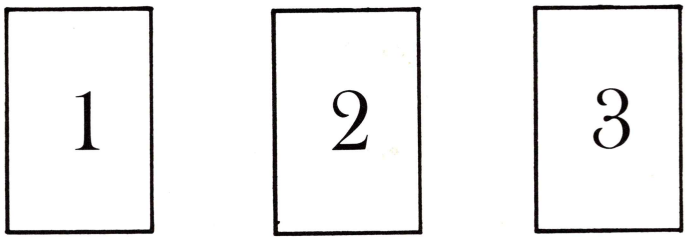
Magic squares have intrigued puzzlers for centuries. Merlin has an interesting problem along these lines to test you on. Below is pictured a four-sided magic square with four of the numbers already filled in (Merlin was in a generous mood when he handed me this puzzle). The object is to fill in the remaining empty squares with the following numbers: 4, 5, 6, 8, 10, 11, 12, 13, 15, 18, 19, and 20. When all of the numbers are correctly in place the magic square will total 47 in every direction, horizontally, vertically, and diagonally. Also, the four courner squares will total 47, and, so will the four inner squares. This problem should keep you busy for a while.

	14		
17			
			9
		7	

A Folding Puzzle

Here's another problem I think will keep you all busy for a while. In the illustration below we see a piece of 8½″ by 11″ typewriter paper folded along line AB. Using the three lengths indicated on the drawing, can you calculate the length of this folded line, AB? (Don't try measuring the picture, it's drawn slightly out of proportion).



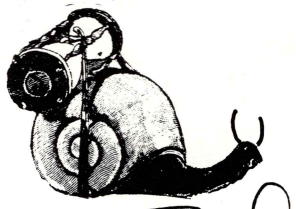


A Square Deal

Pictured above are three face down playing cards. Your job is to name these cards using the following clues to guide you. (1) There is at least one six just to the right of a four. (2) There is at least one six just to the left of a six. (3) There is at least one heart just to the left of a spade. (4) There is at least one heart just to the right of a heart.

Let's see how good you are at playing this hand.

Answers on page 47.

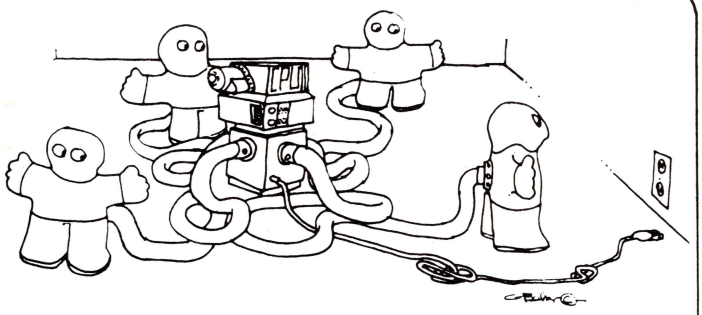


A Snail's Pace

This is a fine old problem. An ambitious snail that wants to get ahead in life starts at the bottom of a well 16 feet deep and begins climbing towards the top. Every day he crawls up 4 feet and every night he slips back down 3 feet. He's one of those snails that never give up, and eventually he got out of the well. How long did it take this snail to win his freedom.

I hope that you have enjoyed the problems presented here. Until we meet again, Merlin and I wish all of you...good puzzling!

Your editor, Charles Barry Townsend



The Best Medicine

Your program is finally finished. You type RUN. Your computer replies READY. It has just eaten five hours of your life. What do you do?

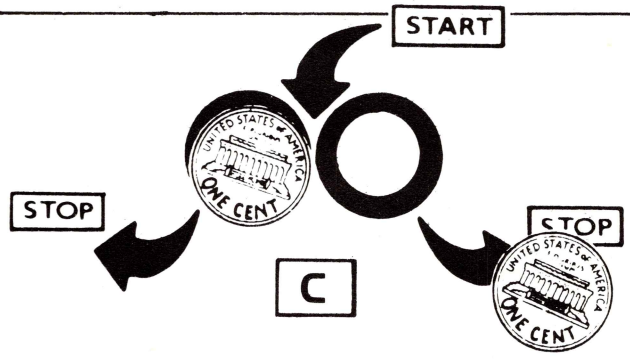
Try to maintain your sense of humor—reach for *The Colossal Computer Cartoon Book*. Laugh at the original adventures of Edu-Man. Chuckle at cartoons by *Creative Computing* favorites Sandy Dean, Harbaugh, Swan, and Johsn. Smile sympathetically at the ways others have suggested to get even with the infernal machines.

The Colossal Computer Cartoon Book contains hundreds of clever cartoons to tickle the funny bone of anyone who has ever loved or hated a computer.

To order one for yourself or a friend, send $4.95 plus $2.00 for postage and handling to Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950. Credit card buyers call toll-free, **(800)631-8112**; in New Jersey, (201)540-0445.

creative computing

Morris Plains, NJ 07950
Toll-free **800-631-8112**
(In NJ 201-540-0445)



The Root of All Evil

Playing with money can get you into trouble, all right. But it can also teach you and your children what happens inside a computer.

By sliding and flipping pennies (affluent readers can use dimes) you learn exactly how simple computer circuits work.

The first half of *Computer Coin Games* provides directions and diagrams for a variety of games which can be played by anyone—computer enthusiast or not. The second half of the book explains how the games relate to computers.

Computer Coin Games is an inexpensive, entertaining way to introduce children and adults to binary numbers, flip flops and counters. Order your copy today.

Send $3.95 plus $2.00 for postage and handling to Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950. Credit card buyers call toll-free, **(800)631-8112**; in New Jersey, (201)540-0445.

creative computing

Morris Plains, NJ 07950
Toll-free **800-631-8112**
(In NJ 201-540-0445)

---

7 Industry Leaders Offer Their Views

The Future of Personal Computers

How will personal computers change our lives in the future? How will the equipment and its applications evolve in the coming years? What roles will personal computers have in society?

The world's leading authorities on personal computers provided some insightful answers at The Boston Computer Society second annual Forum on the Future of Personal Computers, October 15, 1981.

• **Philip D. Estridge**. Director, Entry Systems Business, International Business Machines, Boca Raton, Florida. Mr. Estridge—the creator of IBM's new personal computer—looked into the near future and "The Next Steps for Personal Computers."

• **H.E. James Finke**. President, Commodore International, Ltd., Norristown, Pennsylvania. Mr. Finke gave his perspective on the explosive growth of microcomputers with "The Mass Market Micro: The Future Ain't What It Used to Be."

• **William H. Gates**. President, Microsoft, Bellevue, Washington. Mr. Gates—the father of microcomputer software—provided an inside look at "Things to Come in Personal Computer Software."

• **A.C. (Mike) Markkula**. President, Apple Computer Inc., Cupertino, California. Mr. Markkula examined forthcoming breakthroughs in personal computer technology in his talk "Making Computers Easier to Use: Trends in the User Interface."

• **Peter Rosenthal**. Marketing Manager, Atari Computer Division, Sunnyvale, California. Mr. Rosenthal offered a vision of "The Home Computer of the Future" and its impact on our homes.

• **Jon Shirley**. Vice President, Radio Shack Computer Merchandising, Fort Worth, Texas. Mr. Shirley explored the business applications of future computers with "Personal Computers in the Office of the Future."

• **Nigel Searle**. Vice President, Sinclair Research Ltd., Cambridge, England. Mr. Searle considered the impact of personal computers on consumers in his talk "The Consumer Marketplace for Future Personal Computers."

Moderated by Jonathan Rotenberg, President, The Boston Computer Society.

All seven presentations along with questions and answers are available on two C-90 tape cassettes (2-1/2 hours) for $25 postpaid. If you would also like to subscribe to *Small Business Computers*, add $12 to your order ($37 total). Or, to subscribe to *Creative Computing*, add $20 ($45 total). Send payment or credit card number and expiration date (Visa, MasterCard, American Express) to the address below or call our toll-free number.

creative computing

39 E. Hanover Avenue
Morris Plains, NJ 07950
Toll-free **800-631-8112**
In NJ 201-540-0445

8K ROM
1K RAM

Part 2

Understanding Floating-point Arithmetic

Dr. Ian Logan

In the last article I described how decimal numbers are converted to their floating-point form for the 8K ROM monitor program, but I did not discuss how floating-point numbers are then manipulated. In this article I shall describe the workings on the 'third language' of the 8K ROM—the CALCULATOR LANGUAGE—that is used to add, subtract, multiply, divide and generally manipulate floating-point numbers.

The CALCULATOR LANGUAGE

This language is a STACK OPERATING language and is used to put numbers on to the calculator stack of the ZX80/81, remove numbers from the calculator stack, perform operations on the numbers that are already on the stack and make safety copies of numbers by copying them to the calculator's memory area.

Figure 1.

Address	HEX. code	Mnemonic	Comment
OAEF	ED	RST 0028,FP-CALC.	i.e. CALL 0028
OAFO	01	DEFB +01	Operation '1'
OAF1	34	DEFB +34	Operation '52'

As an example of how the language is used, the first occurrence of the language is shown in full from the listing of the 8K program in Figure 1. This can be shown by entering the Basic line:

```
PRINT PEEK 2799,,PEEK 2800,,PEEK 2801
```

which gives:

```
239
1
52
```

This example shows that the first step required is to call the calculator subroutine. This is done using an indirect jump through the RESTART at 0028. The actual calculator subroutine occurs at 199C in the 'unimproved' ROM and at 199D in the 'improved' ROM.

The bytes of data that follow an RST 0028 instruction are the operation codes of the CALCULATOR LANGUAGE. In this example the first operation to be performed is operation '1' that asks the calculator to EXCHANGE the top two numbers on the calculator stack. The second operation is operation '52.' This is the END-CALC. operation which shows the calculator that there are no further manipulations to be performed.

In the 8K ROM program there are about 40 calls to the calculator subroutine and practically all of them are a good deal more complicated than this first example from the PRINT command routine.

Dr. Ian Logan, 24 Nurses Lane, Skellingthorpe, Lincoln LN6 OTT.

A Demonstration Program

The only way to understand well just how a STACK OPERATING language is used is to produce a simple demonstration program and then as experience is gained to make the program more ambitious.

Program 1—*A Basic Stack Calculator* is such a program written for a 1K ZX81. The first part of the program—lines 10-70—creates a small 'calculator stack' that appears on the screen as a stack of 5 locations each capable of holding a single decimal number. The very important 'last location' pointer is then shown. This pointer tells the user which location is currently the holder of the 'last value,' i.e., the location that holds the topmost number of the 'current stack.'

There are many points that can be made about stack systems but perhaps the most important point is that numbers are not removed from a stack but always copied from it and with each 'removal' the actions involved are a copying of the 'last value' and the decreasing of the STACK POINTER, in this case, the 'last location' pointer.

In lines 100-130 the program awaits the user's entry and then jumps accordingly. In this first program there are 5 operations that can be performed:

1) Input a number: The user is asked for a number and this number goes to the top of the stack.

2) Print a number: The 'last value' is displayed and the program pauses for a key to be pressed before proceeding. Printing the 'last value' does 'remove' it from the stack and hence loses the value. Any attempt to print a 'last value' from an empty stack leads to failure.

3) Adding two numbers: When there are at least two numbers on the stack, the last two numbers can be added together. This operation has the effect of losing the second operand.

4) Subtracting two numbers: The difference between the last two numbers becomes the 'last value.'

5) Multiplying two numbers: The product of the last two numbers becomes the 'last value.'

Program 1 shows that a stack of 5 locations and the handling of 5 operations can be performed on a 1K machine but the reader is urged to add extra operations if extra RAM is available, or to replace the existing 5 operations if not. Suitable operations to be added are EXCHANGE, DELETE, DUPLICATE, SQR, ** and SIN.

Some examples for a beginner to try are:

a) Print the sum of 2.55 and 3.66. This will involve selecting operation '1' and entering the value 2.55, selecting operation '1' and entering the value 3.66, selecting operation '3' to add the two numbers and finally to select operation '2' to print the result.

b) Print the product of 2.55 and 3.66. This will involve the sames step as before but operation '5' is used instead of operation '3'.

It is customary to describe the steps involved in what is termed 'Reverse Polish Notation' in which commas are used to separate the operations. In the present cases the results would be:

For
PRINT the result of 2.55 + 3.66'
write
STACK 2.55, STACK 3.66, ADD, PRINT'
or more simply
'2.55,3.66,+'.

For
'PRINT the result of 2.55*3.66'
write
'2.55,3.66,*'.

The reader is urged to try his own, more complicated, examples before proceeding to consider the operations of the 8K ROM's CALCULATOR LANGUAGE.

The OPERATIONS of the CALCULATOR LANGUAGE

So far I have described the CALCULATOR LANGUAGE as dealing only with floating-point numbers but this is not the whole answer as the language also manipulates character strings. It does this by considering, when required, the 'last value' as being a set of parameters that describes the string—the second and third bytes hold the pointer to the start of the string and the fourth and fifth bytes the counter for its length. (The first byte is redundant.)

The actual operations are: (operation codes in decimal)

0- Jump if 'last value' is true.
1- Exchange the top two values.
2- Delete the 'last value.'
3- Subtract the 'last value' from the value beneath it.
4- Multiply the top two values.
5- Divide the first value by the 'last value.'
6- Raise the first value to the power of the 'last value.'
7- Logically 'OR' the top two numbers.
8- Logically 'AND' the top two numbers.
9- Test the top two numbers for '<='
10- Test the top two numbers for '>='
11- Test the top two numbers for '<>'
12- Test the top two numbers for '>'
13- Test the top two numbers for '<'
14- Test the top two numbers for '='
15- Add the last two numbers together.
16- Logically 'AND' a string and a number.
17- Test the top two strings for '<='
18- Test the top two strings for '>='
19- Test the top two strings for '<>'
20- Test the top two strings for '>'
21- Test the top two strings for '<'
22- Test the top two strings for '='
23- Concatenate the top two strings.

Note: All the above operations (except operation '0') are binary operations and require two operands. The result 'overwrites' the first operand and this creates a 'new last value,' the second operand is thereby 'deleted.'

Note: A logical operation will, if the result is FALSE, return a 'last value' of zero and, if TRUE, a 'last value' of not-zero, usually one.

24- Perform the 'unary minus' operation.
25- Perform the 'CODE' function.
26- Perform the 'VAL ' function.
27- Perform the 'LEN ' function.
28- Perform the 'SIN ' function.
29- Perform the 'COS ' function.
30- Perform the 'TAN ' function.
31- Perform the 'ASN ' function.
32- Perform the 'ACS ' function.
33- Perform the 'ATN ' function.
34- Perform the 'LN ' function. (Note: Decimal 34)
35- Perform the 'EXP ' function.
36- Perform the 'INT ' function.
37- Perform the 'SQR ' function.
38- Perform the 'SGN ' function.
39- Perform the 'ABS ' function.
40- Perform the 'PEEK' function and thereby return the 'last value' equalling the value held in a location.
41- Perform the 'USR ' function and return with the 'last value' equalling the value of the BC register pair.
42- Perform the 'STR$' function.
43- Perform the 'CHR$' function.
44- Perform the 'NOT ' function.

Note: All the above functions overwrite the 'last value' with the result, i.e., they modify the 'last value' as required.

45- The 'last value' is simply duplicated.
46- A MODULUS operation. The first operand, N, is overwritten with the 'remainder,' and the 'last value,' M, is overwritten with the 'quotient,' INT (N/M).
47- A simple jump.
48- A STACK DATA operation. The following 2-5 bytes are treated as data bytes, expanded and passed to the stack.
49- A DECREASE THE COUNTER operation. The system variable BERG can be used as a counter and this operation in effect performs a DJNZ instruction.
50- Jump if 'last value' is negative.
51- Jump if 'last value' is positive.
52- The END-CALC. operation that forces an EXIT from the calculator subroutine.
53- A REDUCE ARGUMENT operation to modify the argument of a SIN and a COS.
54- A TRUNCATE operation that modifies the 'last value' to give its 'integer towards zero' result.
55- The special SINGLE OPERATION used by the EXPRESSION EVALUATOR.
56- An E-TO-FP operation in which the first operand is multiplied by 10 to the power of the 'last value.'
134- Call the SERIES GENERATOR passing to it 6 constants.
136- Call the SERIES GENERATOR passing to it 8 constants.
140- Call the SERIES GENERATOR passing to it 12 constants.
160- STACK ZERO. Put the value zero on the stack.
161- STACK ONE. Put the value one on the stack.
162- STACK a HALF. Put the value 1/2 on the stack.
163- STACK PI/2. Put the value PI/2 on the stack.
164- STACK TEN. Put the value ten on the stack.
192- 'ST-MEM-0'. Store the 'last value' in MEM-0.
193- 'ST-MEM-1'. Store the 'last value' in MEM-1.
194- 'ST-MEM-2'. Store the 'last value' in MEM-2.
195- 'ST-MEM-3'. Store the 'last value' in MEM-3.
196- 'ST-MEM-4'. Store the 'last value' in MEM-4.
197- 'ST-MEM-5'. Store the 'last value' in MEM-5.
224- 'GET-MEM-0'. Put the value from MEM-0 on the stack.
225- 'GET-MEM-1'. Put the value from MEM-1 on the stack.
226- 'GET-MEM-2'. Put the value from MEM-2 on the stack.
227- 'GET-MEM-3'. Put the value from MEM-3 on the stack.
228- 'GET-MEM-4'. Put the value from MEM-4 on the stack.
229- 'GET-MEM-5'. Put the value from MEM-5 on the stack.

Note: Operations 48, 160-164, and 224-229 all provide a means of putting additional numbers on the calculator stack.

Note: The printing of a floating-point number is not done by a CALCULATOR operation but by the PRINT A FLOATING-POINT NUMBER subroutine at 15D7—unimproved ROM (15DB—improved ROM)

Note: Storing a 'last value' does not 'delete' it.

'YOUR FIRST WORDS'

In order to write machine code subroutines that use the CALCULATOR LANGUAGE it is simply a matter of following the rules:

a) Call the calculator.
b) Stack your numbers.
c) Manipulate them.
d) Exit from the calculator.

Program 2—*Producing A Constant* shows a very simple operation being performed. As it is written, the operation 160 is used to make the 'last value' on the stack a zero and this zero is subsequently printed.

The numbers that are available as constants are limited and therefore an alternative method of passing values from Basic to the calculator stack is needed. Program 3—*Stack Any Number* is therefore the next stage to be reached. This program shows how numbers can be transferred to the stack via the calculator's memory area. In the program the 5 bytes of the floating-point form are transferred to MEM-O and then in the CALCULATOR PROGRAM the number is copied from MEM-O by using operation 224.

Overall View

In this article I have tried to show the essential features of the CALCULATOR LANGUAGE, but I have kept away from discussing how the calculator subroutine might be used. However, it would be useful to include a slightly more complicated example from the 8K program.

The function TAN modifies the value of the 'last value' so that it goes from say, X to TAN X. In the 8K ROM program the actual subroutine is situated at 1D6D—unimproved ROM (16DE—improved ROM). See Figure 2.

Figure 2.

RST 0028	- With X as a 'last value' call the calculator. Stack holds X.
DEFB +2D	- Operation 45 and the 'last value' is duplicated. Stack holds X, X.
DEFB +1C	- Operation 28 and SIN X is found. Stack holds X, SIN X.
DEFB +01	- Operation 1—exchange the values. Stack holds SIN X, X.
DEFB +1D	- Operation 29 and COS X is found. Stack holds SIN X, COS X.
DEFB +05	- Operation 5. The two values are divided. Stack holds SIN X/COS X and as TAN X = SIN X/COS X always the stack holds only TAN X.
DEFB +34	- Operation 52—leave the calculator.
RET	- Return to the calling subroutine.

THE ZX80 HOME COMPUTER PACKAGE

Programs that every HOME COMPUTER should have:

COMPOSER uses a color overlay to produce a multi-octave keyboard for the creation of electronic music. Compositions of hundreds of notes can be saved on tape for later editing, broadcast to nearby AM radio or TV, or recorded directly into a tape recorder. Changes can be easily made.

ETCH-A-SCREEN

Rapidly paint text and graphics on the screen. Store screen display on tape for later viewing or modification.

ELECTRONIC BILLBOARD

Use your computer as a display center. Displays your message in giant letters which move continuously across the screen. Save messages on tape.

CHECKBOOK BALANCER

Keep a running tabulation of your bank account. Reconciles bank statement to current actual balance, and displays both. Stores and displays up to 30 uncleared monthly transactions.

CALCULATOR

Give your computer high-precision floating point arithmetic. Multiplies or divides two numbers ranging from .0000000001 to 9999999999.

THE ZX80 HOME COMPUTER PACKAGE contains: manual, a cassette of programs, two reference cards, two keyboard overlays, a blank score sheet, and a blank SCREEN DISPLAY sheet.

For the ZX 80 & MicroAce with 4K BASIC and 1 K memory or more

$9.95

NO POSTAGE.
NO HANDLING.
NO SALES TAX.

SUPER Z

SUPER Z builds machine-code modules that add seven new statements to ZX80 and MicroAce 4K BASIC: TAB, SCROLL, MEM, PAUSE, READ, RESTORE, and DATA. Most statements are used in the form of a USR function, (like PRINT USR (MEM) which prints the amount of unused memory).

Expand your 4k BASIC with SUPER Z. The SUPER Z package contains a manual, reference cards, and a cassette with the SUPER Z program, a ready-to-use SUPER Z module with all instructions, and a SUPER Z demonstration program. Send check or money order for $9.95 to LAMO-LEM LABS.

SEND FOR OUR CATALOG OF ZX80, MICROACE, APPLE, AND TI 57, 58, & 59 PRODUCTS, INCLUDING FREE ZX80/M.ACE CODING SHEETS.

LAMO-LEM
CODE 205
BOX 2382
LA JOLLA, CA 92038

Any reader who wishes to understand the CALCULATOR LANGUAGE is urged to read:

SINCLAIR ZX81 ROM Disassembly, Part A: 0000H-0F54H, by Dr. Ian Logan, and *Part B: 0F55H-1DFEH*, by Dr. Ian Logan and Dr. Frank O'Hara. Part A deals mainly with the operating system whereas Part B deals mainly with the CALCULATOR LANGUAGE as discussed above. The books are published by Melbourne House (Publishers) Ltd.

This month's program is not a game but a program that shows the 'randomness' of RND. (A ZX80 4K ROM program of this type was included in *The ZX80 Companion*)

In the *Random Number Graph* the user is asked to provide a starting seed value for the random number generator and the program then takes the next 200 random numbers for its analysis.

The program produces a 'curve' that shows just how 'non-random' the pseudo-number generator actually is in the 8K ROM.

I would be interested to know if anyone finds a seed value that gives a straight line—remember that the sequence of random numbers can be changed by addition of a dummy call to RND.

Listing 1: A Basic Stack Calculator.

```
  10 DIM A(VAL "5")                    THE STACK HAS 5 LOCATIONS.
  20 LET Z=VAL "40"                    Z IS A DUMMY VARIABLE.
  30 LET P=SGN PI                      P WILL POINT TO THE FIRST
  40 CLS                                  LOCATIONS.
  50 FOR B=VAL "5" TO SGN PI STE
P -SGN PI
  60 PRINT "LOC. ";B;" HOLDS",A(
B)
  70 NEXT B
  80 PRINT AT VAL "8",NOT PI;""""  SHOW WHICH LOCATION WAS THE
LAST"" LOC. IS ";P-SGN PI              "LAST LOCATION".
  90 PRINT AT VAL "13",NOT PI;"E
NTER CODE"
 100 PRINT "(1-IN.,2-PR.,3-ADD,4  ACCEPT THE OPERATION CODE.
-SUB,5-MULT)"
 110 INPUT B
 120 LET P=P-SGN PI                   DECREASE THE POINTER.
 130 GOTO B*VAL "200"                 JUMP ACCORDING TO THE CODE.
     INPUT A NUMBER
 200 PRINT AT VAL "19",NOT PI;"I
NPUT VALUE"
 210 INPUT B                          ACCEPT A VALUE.
 220 LET A(P+SGN PI)=B                ASSIGN THE VALUE.
 230 LET P=P+VAL "2"                  INCREASE THE POINTER.
 240 GOTO Z                           START AGAIN.
     PRINT A NUMBER
 400 PRINT AT VAL "19",NOT PI;"L
AST VALUE = ";A(P)                    PRINT THE "LAST VALUE".
 410 PAUSE Z*Z                        WAIT FOR A KEY TO BE PRESSED.
 420 GOTO Z                           START AGAIN.
     ADD TWO NUMBERS
 600 LET A(P-SGN PI)=A(P-SGN PI)      ADD THE NUMBERS.
+A(P)
 610 GOTO Z                           START AGAIN.
     SUBTRACT TWO NUMBERS
 800 LET A(P-SGN PI)=A(P-SGN PI)      SUBTRACT THE NUMBERS.
-A(P)
 810 GOTO Z                           START AGAIN.
     MULTIPLY TWO NUMBERS
1000 LET A(P-SGN PI)=A(P-SGN PI)      MULTIPLY THE NUMBERS.
*A(P)
1010 GOTO Z                           START AGAIN.
```

Note: In order to get this program working in a 1K machine, the values zero and one have been replaced by NOT PI and SGN PI respectively; all other integers are entered as VAL strings and the dummy variable Z is used. If more than 1K RAM is available, then none of these features need to be included.

Listing 2: Producing a Constant.

```
  5 REM USE THE PRINTER BUFFER
 10 LET A=16444
 15 REM CALL THE CALCULATOR
 20 POKE A,239                        RST 0028
 25 REM CHOOSE AN OPERATION IN
       THE RANGE 160-164.
 30 POKE A+1,160                      OPERATION "160"
 35 REM EXIT FROM CALCULATOR
 40 POKE A+2,52                       OPERATION "52"
 45 REM CALL PRINT-NUMBER
 50 POKE A+3,205                      CALL 15D7 (OR 15DB)
 60 POKE A+4,215                      (USE 219 FOR IMPROVED ROM)
 70 POKE A+5,21
 75 REM EXIT FROM USR
 80 POKE A+6,201                      RET
 85 REM  A SIMPLE HEADER
 90 PRINT "LAST VALUE WAS ";
 95 REM RUN MACHINE CODE
100 RAND USR A
```

Listing 3: Stack Any Number.

```
  5 REM ENTER YOUR NUMBER
 10 INPUT N
 15 REM TRANSFER IT TO MEM-0
 20 LET V=PEEK 16400+256*PEEK 1
6401
 30 FOR B=1 TO 5
 40 POKE 16476+B,PEEK (V+B)
 50 NEXT B
 60 LET A=16444
 70 POKE A,239                    RST 0028
 75 REM CHOOSE YOUR OPERATIONS
NOW AND INCLUDE OPERATION "224"
 80 POKE A+1,224                  OPERATION "224"
 90 POKE A+2,52                   OPERATION "52"
100 POKE A+3,205                  CALL 15D7 (15DB)
110 POKE A+4,215                  (USE 219 FOR IMPROVED ROM)
120 POKE A+5,21
130 POKE A+6,201                  RET
140 PRINT "THE NUMBER WAS ";
150 RAND USR A
```

Note: The transfer of the 5 bytes from the variable area can be done in machine code as shown in Figure 3.

Figure 3.

```
LD      HL,(VARS)
INC     HL
LD      BC,+0005
LD      DE,(STKEND)     Use STKEND for direct transfer to
LDIR                    the stack, or MEMBOT to use MEM-0.
LD      (STKEND),DE     (stacking only)
```

Listing 4: The Random Number Graph Program

```
 10 PRINT AT VAL "18",NOT PI;"E
NTER NUMBER"
 20 INPUT N                       SUPPLY A SEED 0-65535
 30 CLS
 40 PRINT "RANDOM NUMBER GRAPH"
 50 PRINT "===================="
 60 RAND N
 70 DIM A(VAL "10")               THE ARRAYS FOR THE 10 GROUPS
 80 DIM B(VAL "10")
 90 FOR N=SGN PI TO VAL "100"     TAKE 100 CALLS TO RND
100 LET R=SGN PI+INT (VAL "10"*   AND CREATE GROUPS FOR EACH TENTH
RND)                              IE.  .1, .2,.....,9
110 LET A(R)=A(R)+SGN PI          COUNT THE GROUPINGS
120 NEXT N
130 FOR N=SGN PI TO VAL "5"       SHAPE THE GRAPH BY TAKING THE
140 LET K=N                       OUTER PAIR, THE NEXT PAIR,ETC
150 GOSUB VAL "300"
160 LET K=VAL "11"-N
170 GOSUB VAL "300"
180 NEXT N
190 FOR N=SGN PI TO VAL "10"      PRINT THE GRAPH
200 PRINT "+";TAB B(N);"▒"        GRAPHIC SHIFTED A
210 NEXT N
220 PRINT "01234567890123456789   PRINT THE SCALE
"
230 RUN
300 LET M=VAL "50"                SET M AS A UPPER LIMIT
310 FOR J=SGN PI TO VAL "10"      LOOK AT THE 10 SCORES
320 IF M<=A(J) THEN GOTO VAL "3   ASSIGN THE LOWEST TOTAL TO B(K)
50"
330 LET M=A(J)
340 LET L=J
350 NEXT J
360 LET B(K)=A(L)                 SET EACH "LOWEST TOTAL" TO A
370 LET A(L)=VAL "40"             HIGH VALUE, TO SHOW IT HAS
380 RETURN                        BEEN USED
```

RUN—I suggest in FAST.

Note: If more numbers would be preferred, change, e.g., as follows:

For 10,000 random number:

90 FOR N=SGN PI TO VAL "10000"

and

110 LET A(R)=A(R)+VAL ".01"

This gives an almost straight line from a starting seed of 1.

NOW AVAILABLE

keyboards

Standard Computer Keyboard Designed for ZX81, ZX80, & MicroAce

- Same switches used on Apple Computers
- Two shift keys • 6-inch space bar

Plans for keyboard conversion with reverse video — $5.00
Keyboard with complete parts and plans — $55.00
Wired keyboard, complete with plans — $75.00

Shipping Charge (by UPS) — $5.00

SEND CHECK OR MONEY ORDER TO:

L.J.H. Enterprises

P.O. Box 6305, Orange, CA 92667
(714) 547-8717 Visa & M/C Accepted

ZX80 **JRS SOFTWARE** ZX81

19 WAYSIDE AVENUE, WORTHING, SUSSEX, BN13 3JU
TELEPHONE WORTHING 65691 (Evenings and Weekends only)

16K RAM PACK £35 ($69.95) **WHY PAY MORE**

Fully built, tested and guaranteed
Black case
No 'wobble' problems — fully supported by ZX81
Fully compatible with printer etc, etc.
Full refund if not delighted

(Please allow 14-21 days for delivery)

An ESSENTIAL addition to your 1K RAM ZX81 (or ZX80 8K ROM)
(please state which when ordering)

TOOLKIT (written by PAUL HOLMES)

Provides the following additional facilities:

Line renumber — you state starting number and increment value.
Search and replace — changes every occurence of a character as you require.
Free space — tells you how many free bytes you have left

SPECIAL GRAPHICS ROUTINES

Hyper graphics mode — graphics never seen on a ZX81 before.
Open — instantly sets up as many empty print lines as you require. (1K version only)
Fill — used in conjunction with OPEN fills your screen instantly with your specified character
Reverse — changes each character on your screen to its inverse video.

TAPE ROUTINE — provides a system **WAIT** condition until a signal is received in the cassette ear jack — many uses !

All these routines are written in machine code and together take up only 164 BYTES of your precious RAM - an incredible achievement!!

The price is incredible too! ONLY **£3.95 ($7.90)** for cassette, including FULL instructions and example programs.

ALSO available 16K version ONLY **£4.95 ($9.90)** which includes all the above PLUS:
GOTO's and GOSUB's included in line renumber.
Search for and list every line containing specified character. **16K VERSION**

CASSETTE professionally recorded by SOUND NEWS STUDIOS

GAMES PACK — ***Beat this for value! 5 × 16K programs PLUS 2 × 1K programs*** **£4.95 ($9.90)**

3-D Battle (M/code-1K) — Fast-moving space battle with continuous count-down of energy units left.

City Bomb (M/code-1K) — Destroy the buildings and land your plane. Your fuel has nearly gone and you circle the city lower and lower.

Warp Wars (Basic & M/code-16K) — Features realistic space-craft moved by M/code for instant response.
(previously sold at Microfair with Sweet Tooth for £4.95)

Snake (Basic-16K) — A game of thought and skill. Pass through all the marked squares without crossing or doubling back on your path, but watch out for the expanding black blob.
(previously sold at Microfair for £3.95)

Sweet Tooth (Basic & M/code-16K) — M/code routines used to move your fat face round the screen and gobble the sweets.

PLUS Slalom and Black Holes (previously sold together for £4.95)

OVERSEAS CUSTOMERS PLEASE NOTE Payment may be made in Sterling (Money Order available at your bank) or $U.S. (U.S.A. customers only).

ALL GOODS SENT AIRMAIL

ZX 81 GAMES

FED UP WITH BEING RIPPED OFF?
HAVE YOU BROUGHT
BORING/WORTHLESS/RUBBISH GAMES?
DON'T DESPAIR, TRY THIS!

GAMESTAPE 1, for 1K **only £3.95**
10 Games incl. ASTEROIDS, UFO, CODE, BOMBER GUILLOTINE, etc
PROBABLY THE BEST VALUE 1K TAPE AVAILABLE!

GAMESTAPE 2, for 16K **only £4.95**
*STARFIGHTER... You are fighting at the end of the Universe, how many Enemy Craft can you destroy, before your energy runs out!
PYRAMID... Can you move the PYRAMID? one mistake and it will collapse! A Thinkers Game.
ARTIST... Draw on the screen, then use the 10 Memories to Store your drawings. Incl. SAVE, COPY, etc.

GAMESTAPE 3, for 16K **only £5.95**
*CATACOMBS... A Multi-Level Graphics Adventure. You are alone and lost in the CATACOMBS, how much Gold can you find? How long can you survive, before you starve to death, or one of the many Monsters gets you!
Strange things can happen, but it's up to you to discover the Secrets of the CATACOMBS!

GAMESTAPE 4, for 16K **only £5.95**
*3D MONSTER MAZE... UNBELIEVABLE GRAPHICS.
Can you find your way through the Maze? The Exit is there somewhere, but then so is a T.Rex, and it's after YOU!
ALL IN 3D, YOU'VE NEVER SEEN ANYTHING LIKE THIS BEFORE!

THE FINEST QUALITY SOFTWARE AVAILABLE
(ABSOLUTELY NO RUBBISH)
Games Marked * incl. Machine Code.

DEPT C. Cheque/P.O.s to
J.K. GREYE SOFTWARE
16 PARK STRET, BATH, AVON, BA1 2TE

Orders outside UK- Send payment in Sterling & incl. appropriate postage (i.e. US Air Mail — add£1.50).

TAKE YOUR 1K RAM ZX81 COMPUTER TO THE LIMIT

with ZETAPAKs...6-packs of programs on cassette tape with instructions and author's notes. All in BASIC.

ZETAPAK #1: War Games $9.95
Bomb Run, Sub Hunter, Barrage, Fighter Pilot™, Torpedo Alley™, and Dragon's Teeth (tank battle)™

ZETAPAK #2: Basically BASIC $9.95
RaNDom POKEr, RETURN Address, LOOPer, IF I May, InVARIABLEly DisARRAYed, and PAUSE

ZETAPAK #3: SciFi Fantasy $9.95
Planetfall™, Turret Laser, Arcade Invaders, Hyperspace Display, and Star Catcher

Order ZETAPAKs by number and name or "Special 3" for all 3 at $24.95

TO ORDER

Specify pack(s) wanted, enclose payment + $2 S& H, and mail to:
ZETA Software
P.O. Box 3522
Greenville, S.C. 29608 - 3522
(Please allow 4 weeks for delivery).

Or write for catalog of 40+ titles including 8K/16K scientific programs.

ζζζζ ζζζζ ζζζζζ ζζ
ζ ζ ζ ζ ζ
ζ ζζζζ ζ ζζζζ
ζζζζ ζζζζ ζ ζ ζ

In Europe, send $1 bill or DM2 in stamps to:
DELTASOFT
Dr. Walter Diembeck
Osterfeidstr. 79d
D-2000 Hamburg 54
Germany

The $149.95 personal computer.

Introducing the Sinclair ZX81

If you're ever going to buy a personal computer, now is the time to do it.

The new Sinclair ZX81 is the most powerful, yet easy-to-use computer ever offered for anywhere near the price: only $149.95* completely assembled.

Don't let the price fool you. The ZX81 has just about everything you could ask for in a personal computer.

A breakthrough in personal computers

The ZX81 is a major advance over the original Sinclair ZX80—the world's largest selling personal computer and the first for under $200.

In fact, the ZX81's new 8K Extended BASIC offers features found only on computers costing two or three times as much.

Just look at what you get:

- Continuous display, including moving graphics
- Multi-dimensional string and numerical arrays
- Mathematical and scientific functions accurate to 8 decimal places
- Unique one-touch entry of key words like PRINT, RUN and LIST
- Automatic syntax error detection and easy editing
- Randomize function useful for both games and serious applications
- Built-in interface for ZX Printer
- 1K of memory expandable to 16K

The ZX81 is also very convenient to use. It hooks up to any television set to produce a clear 32-column by 24-line display. And you can use a regular cassette recorder to store and recall programs by name.

*Plus shipping and handling. Price includes connectors for TV and cassette, AC adaptor, and FREE manual.

If you already own a ZX80

The 8K Extended BASIC chip used in the ZX81 is available as a plug-in replacement for your ZX80 for only $39.95, plus shipping and handling—complete with new keyboard overlay and the ZX81 manual.

So in just a few minutes, with no special skills or tools required, you can upgrade your ZX80 to have all the powerful features of the ZX81. (You'll have everything except continuous display, but you can still use the PAUSE and SCROLL commands to get moving graphics.)

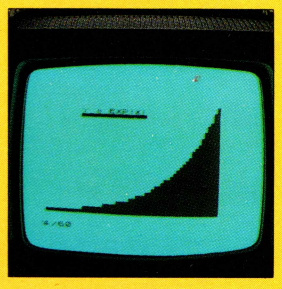
With the 8K BASIC chip, your ZX80 will also be equipped to use the ZX Printer and Sinclair software.

Order at no risk**

We'll give you 10 days to try out the ZX81. If you're not completely satisfied, just return it to Sinclair Research and we'll give you a full refund.

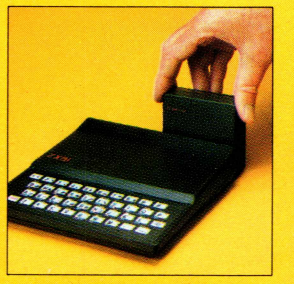
And if you have a problem with your ZX81, send it to Sinclair Research within 90 days and we'll repair or replace it at no charge.

**Does not apply to ZX81 kits.

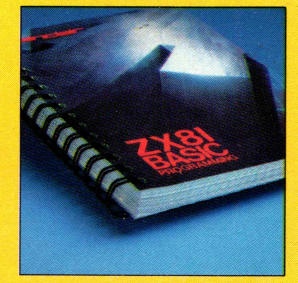


NEW SOFTWARE: Sinclair has published pre-recorded programs on cassettes for your ZX81, or ZX80 with 8K BASIC. We're constantly coming out with new programs, so we'll send you our latest software catalog with your computer.

ZX PRINTER: The Sinclair ZX Printer will work with your ZX81, or ZX80 with 8K BASIC. It will be available in the near future and will cost less than $100.



16K MEMORY MODULE: Like any powerful, full fledged computer, the ZX81 is expandable. Sinclair's 16K memory module plugs right onto the back of your ZX81 (or ZX80, with or without 8K BASIC). Cost is $99.95, plus shipping and handling.



ZX81 MANUAL: The ZX81 comes with a comprehensive 164-page programming guide and operating manual designed for both beginners and experienced computer users. A $10.95 value, it's yours free with the ZX81.

The $99.95 personal computer.

Introducing the ZX81 kit

If you really want to save money, and you enjoy building electronic kits, you can order the ZX81 in kit form for the incredible price of just $99.95.* It's the same, full-featured computer, only you put it together yourself. We'll send complete, easy-to-follow instructions on how you can assemble your ZX81 in just a few hours. All you have to supply is the soldering iron.

How to order

Sinclair Research is the world's largest manufacturer of personal computers.

The ZX81 represents the latest technology in microelectronics, and it picks up right where the ZX80 left off. Thousands are selling every week.

We urge you to place your order for the new ZX81 today. The sooner you order, the sooner you can start enjoying your own computer.

To order, simply call our toll free number, and use your MasterCard or VISA.

To order by mail, please use the coupon. And send your check or money order. We regret that we cannot accept purchase orders or C.O.D.'s.

CALL 800-543-3000. Ask for operator #509. In Ohio call 800-582-1364. In Canada call 513-729-4300. Ask for operator #509. Phones open 24 hours a day, 7 days a week. Have your MasterCard or VISA ready.

These numbers are for orders only. For information, you must write to Sinclair Research Ltd., 2 Sinclair Plaza, Nashua, NH 03061.

sinclair

AD CODE 03SY

	PRICE†	QTY.	AMOUNT
ZX81	$149.95		
ZX81 Kit	99.95		
8K BASIC chip (for ZX80)	39.95		
16K Memory Module (for ZX81 or ZX80)	99.95		
Shipping and Handling	4.95		$4.95
		TOTAL	

MAIL TO: Sinclair Research Ltd., One Sinclair Plaza, Nashua, NH 03061.

NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

† U.S. Dollars

4K ROM
1K RAM

Plotting with 4K Basic

Karl Brendel

Have you tried plotting with the 4K Basic? Or experimented with user-input functions? The 4K Basic does not seem well suited to those purposes, does it? For many plotting applications, resolution appears limited to the size of the "inverse space." Functions being used repeatedly need to be in the body of the program; changing them with each run is a source of bugs.

Of course, if you have the 8K Basic, your problems are largely solved by using PLOT, PRINT AT and VAL. But how would you have them with 4K Basic?

If you have only 1K of RAm, this program provides you with a field of 10 lines by 7 columns for plotting. (More RAM? See the notes at the end of this article.) Pixel graphics provide 280 plotting positions: the field is 20 pixels high by 14 pixels wide.

After loading the program, key GOTO 30, NEWLINE. The plotting field will appear, with the prompt XM=. Supply the starting value for X (the independent variable); key NEWLINE. The prompt XS= will appear. Supply numerical value which you want to represent by one pixel in the horizontal direction. (Remember that there are 14 pixels to the line.) Key NEWLINE. The prompt YS= will appear. Enter the numerical value which you want to represent by one pixel in the vertical direction (with 20 pixels to the column); key NEWLINE.

At last the prompt Y= will appear, with the cursor between quotation marks. Enter the expression you wish to plot. For example, if you want to plot a parabola, your entry might be X**2-96. (Your expression is limited to ten characters and can include both arithmetic and logical operators. Of course, integer math will govern the outcome of your calculations.) End the expression with NEWLINE. After a pause of about two seconds, your plot will appear.

Karl Brendel, Sunflower Systems, 16573 E. Purdue Pl., Aurora, CO 80013.

After one expression has been plotted, you will see the cursor inside the quotation marks. You will not see the expression printed on the screen. (This is to avoid out-of-screen and out-of-memory errors.) Your response to the cursor depends on your choice from among three options:

a) To overlay the plot of another expression on the existing plot, using the current values of XM, XS and YS: Key M (More), NEWLINE, then the new expression and NEWLINE.

b) To produce a clean field for a new expression: Key N (New), NEWLINE, then proceed as before.

c) To exit the program: Key NEWLINE without M or N.

The line at the middle of the field represents the line Y=0 (the "X axis"). That does not actually limit the range of Y values you can examine. For example, suppose you want to plot Y=X**2. All of the Y values will be positive. However, you do not have to waste the lower half of the field. You can use its full height by subtracting a number from Y. Suppose XM=-6 and XS=1. Then the maximum X will be +7. The greatest value of Y will be +49, the least, 0. A value of YS=6 would make the plot fit the field, but use only the upper half. Instead, plot Y=X**2-25. Now you can use YS=3 to spread your plot some, using the entire field. Just remember that your plot has been arbitrarily shifted down by 25.

If you try to enter an expression longer than ten characters, the program will return to the prompt without attempting to plot.

Values of Y outside the range of the field will not blow the program—they will just be skipped. Operations which cause overflow and illegal operations will stop the program.

The three major challenges to constructing the program were:

a) to create the field, reserving space in the display file;

b) to process the user-input expression without returning to the list;

c) to determine and plot the correct ZX80 character.

Creating the Field

The field is essentially a rectangle formed of spaces, with a line of shifted W's for the X axis. In order to calculate plot positions within the field, I wanted the lines to be of known and constant length. The solution was to print a series of spaces: four lines of seven spaces, one line of seven shifted T's, and five lines of seven spaces.

After the field routine ran well in Basic, I rewrote it in Z80 machine language, using a program by Dr. I. S. Logan (*SYNC* 1:2) as my model (see Listing 2). The advantages were conservation of program memory (leaving more memory for the display file) and improved speed of execution; about two seconds in Basic became "instantaneous" in machine language.

(Finding that I could program the ZX80 in machine language led me to investigate the subject further. For the interested reader, I would strongly urge the purchase of *How to Program the Z80* by Rodney Zaks.)

Processing the Expression

When input, the expression is stored as Y$. I reserved space for a numerical copy of the expression in line 10, which is initialized as LET Y=000000000000. Line 20 is a RETURN, allowing the expression to be evaluated repeatedly by the command GOSUB 10.

Lines 200-270 accomplish the conversion of the expression from a string variable to a program line. CODE(Y$) determines the first character in the expression. That character is then POKEd into the location originally occupied by the first 0 in line 10. TL$(Y$) then removes the first character of Y$. The operation is repeated until the entire expression has been POKEd into line 10, or until an eleventh character is detected. If the final character in the expression is the tenth, or lower, then a "+" is POKEd into the next location in line 10, to provide a mathematically correct expression.

Would your believe that this program modifies itself? For example, after processing Y=X**2-25, line 10 will read
10 LET Y=X**2-25+00000.
While this technique may be valuable in other applications, you should observe the following cautions: Be *sure* that you modify only the portion which you intend to. Be *very* sure that your documentation (REMARKs, user notes, etc.) states clearly what you have done, and what the purpose is. Otherwise, someone else using your program may become very confused by such changes in the program listing, especially if unexplained errors have occurred during a run. Even you might forget just what you had intended.

The program is modified again, to the initialized state, with another machine language routine which restores the 0's to line 10. See Listing 3. This is done to allow the More and New options, while still keeping the user out of the list. I considered using a Basic FOR-TO-NEXT loop, or setting lines 200-270 apart as a subroutine which could then have processed Y$="000000000000".
In each case, the memory "overhead" seemed too high for 1K: The subroutine approach would have used an additional 38 bytes; the machine language routine uses only 11 bytes. (An additional four bytes are used by the Basic REMARK line in which the routine is stored. This was done for the sake of clarity; the 0-POKEing routine could have been stored in the same line as the field routine.)

Determining the Character

Given the field coordinates (row and column) of the pixel to be plotted, the problem was to determine which character (if any) occupied those coordinates, and replace it with a character containing the required pixel and any existing pixel.

The ZX80 character set provides characters which represent all possible combination of pixels grouped two by two—from the "space" (nothing plotted) to the "inverse space" (all four locations plotted). Since each pixel in a group can be only dark or light, it is easy to view the group as a four digit binary number: a 0 represents a light pixel and a 1 represents a dark one. I did that and coded the possible group configurations that way. See Figure 1.

Figure 1.

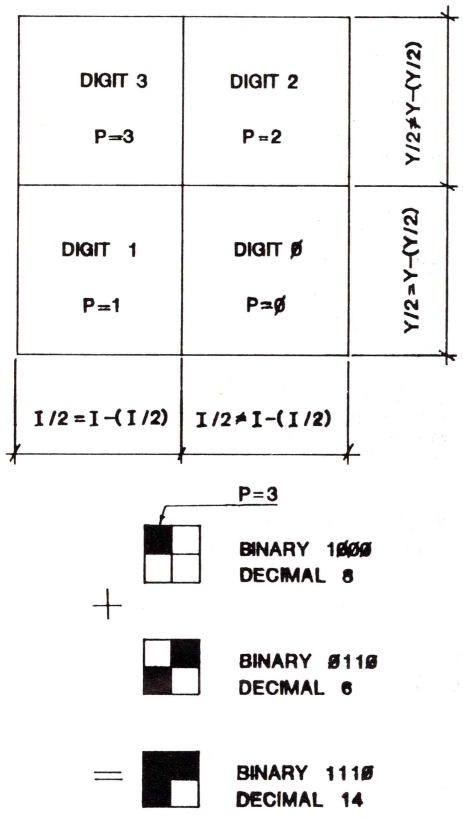


Figure 2.

CHARACTER	D(3)	D(2)	D(1)	D(Ø)	J = DECIMAL OF BINARY CODE	C(J) = ZX-8Ø CODE
□	Ø	Ø	Ø	Ø	Ø	Ø
▘	Ø	Ø	Ø	1	1	7
▝	Ø	Ø	1	Ø	2	6
▀	Ø	Ø	1	1	3	3
▖	Ø	1	Ø	Ø	4	5
▌	Ø	1	Ø	1	5	13Ø
▞	Ø	1	1	Ø	6	8
▛	Ø	1	1	1	7	132
▗	1	Ø	Ø	Ø	8	4
▚	1	Ø	Ø	1	9	136
▐	1	Ø	1	Ø	1Ø	2
▜	1	Ø	1	1	11	133
▄	1	1	Ø	Ø	12	131
▙	1	1	Ø	1	13	134
▟	1	1	1	Ø	14	135
■	1	1	1	1	15	128

Unfortunately, the ZX80 character set does not use such a transparent system of values. In order to translate the ZX80 codes into my system of binary codes, I took the following approach:

I created arrays C (with 16 elements) and D (with 4 elements). D is used to hold the binary code for a particular group: Each element of D is either a 1 or a 0 depending on the pixel it represents. C is loaded with ZX80 character codes in such a fashion that C(J) contains the ZX80 code of the character which has a binary code equal to (decimal) J. For example, C(0)=0, the space; C(15)=128, the inverse space. See Figure 2.

The values of I (equivalent to X/XS) and Y are first converted into field rows and columns. Then a PEEK is done at the corresponding location in the display file to determine which character is currently at that location. The ZX80 code for that character is matched against the elements of C until the correct match is found, or until all elements have been compared. If a match is found, the value of J at that point is taken as the decimal value of the binary code for the character. If no match can be made, the character is treated as a space, and J is set equal to zero.

Lines 540-570 convert the decimal value of J into binary, loading the elements of D with either 1's or 0's as appropriate. This is done by a series of (integer) divisions, which have the additional use of leaving J set equal to zero. Then the element of D which corresponds to the pixel to be plotted, D(P), is set equal to one, effectively making that pixel dark, regardless of its previous state.

Lines 600-620 convert the elements of D back into a decimal value, again called J. Element C(J) is now the ZX80 code for the desired character to be plotted, and it is POKEd into the same location where the PEEK was done.

When you study the program listing, you will not see any lines which DIMENSION C and D, or any which give the values to the elements of C. These operations take place during the original entering and saving of the program. Listing 4 shows a program which will create C and D and provide the proper values of C. This program should be entered and run before the plotting program is entered. C and D

Listing 1

```
 10 LET Y=000000000000
 20 RETURN
 30 LET I=USR(16427)
100 PRINT "XM=";
110 INPUT XM
120 PRINT XM,"XS=";
130 INPUT XS
140 PRINT XS,"YS=";
150 INPUT YS
160 PRINT YS,"Y="
170 INPUT Y$
180 LET I=USR(16495)
200 LET I=16512
210 IF Y$="" THEN GO TO 270
220 IF I=16522 THEN GO TO 170
230 POKE I,CODE(Y$)
240 LET I=I+1
250 LET Y$=TL$(Y$)
260 GO TO 210
270 POKE I,19
300 FOR I=0 to 13
310 LET X=XM+I*XS
320 GOSUB 10
330 LET Y=Y/YS
340 IF Y<-10 OR Y>9 THEN GO TO 640
400 LET P=0
410 IF I/2=I-I/2 THEN LET P=1
420 IF NOT Y/2=Y-Y/2 THEN LET P
=P+2
430 LET X=I/2+1
440 LET Y=(9-Y)/2
450 LET C=PEEK(256*PEEK(16397)+
    PEEK(16396)+8*Y+X)
500 FOR J=0 TO 15
510 IF C(J)=C THEN GO TO 540
520 NEXT J
530 LET J=0
540 FOR K=0 TO 3
550 LET D(K)=J-2*(J/2)
560 LET J=J/2
570 NEXT K
580 LET D(P)=1
600 FOR K=0 TO 3
610 LET J=J+D(K)*2**K
620 NEXT K
630 POKE 256*PEEK(16397)+PEEK
    (16396)+8*Y+X,C(J)
640 NEXT I
700 INPUT Y$
710 IF Y$="N" THEN GO TO 30
720 IF Y$="M" THEN GO TO 170
```

SYNCSUM= 59 (As the listing is shown—without REM lines 1 and 2 or any machine language routines.)

Listing 2

USR(16427)

```
     LD HL, (DFILE)        42, 12, 64    Locate start of Display File.
     INC HL                35            Point to first column, first line.
     LD B, #4 (#9)         6, 4 (9)      Set up for top 4 (9) lines.
(1)  PUSH BC               197           Save line tally (B).
     CALL SUB I            205, 94, 64   Create a line.
     POP BC                193           Recover line tally.
     DJNZ (1)              16, 249       Repeat for 4 (9) lines.
     LD B, #7 (#32)        6, 7 (32)     Prepare to draw X axis.
(2)  LD (HL), #3           54, 3         Poke code of X axis character (CHR$(3)).
     INC HL                35            Point to next column.
     DJNZ (2)              16, 251       Repeat for 7 (32) columns.
     LD A, #117            62, 117       Prepare for "NEWLINE".
     INC A                 60            Increase A to 118 (="NEWLINE").
     LD (HL),A             119           POKE "Newline" at end of line.
     INC HL                35            Point to first column, next line.
     LD B, #5, (#10)       6, 5 (10)     Set up for lower 5 (10) lines.
(3)  PUSH BC               197           Save line tally.
     CALL SUB I            205, 94, 64   Create a line.
     POP BC                193           Recover line tally.
     DJNZ (3)              16, 249       Repeat for 5 (10) lines.
     LD DF-EA, HL          34, 14, 64    Tell System where field ends.
     LD DF-END, HL         34, 16, 64
     LD A, #14 (#4)        62, 14 (4)    Set up System Line Counter.
     LD LINE COUNTER, A    50, 37, 64
     LD A, #0              62, 0         Set up System Space Counter.
     LD SPACE COUNTER, A   50, 36,64
     RETURN                201
(1)  LD B, #7 (#32)        6, 7 (32)     Set up for 7 (32) columns.
(4)  LD (HL), #0           54,0          POKE a "space".
     INC HL                35            Point to next column.
     DJNZ (4)              16, 251       Repeat 7 (32) times.
     LD A, #117            62, 117       Prepare for "NEWLINE".
     INC A                 60            Put 118 in Accumulator.
     LD (HL), A            119           POKE "NEWLINE" at end of line.
     INC HL                35            Point to the first column, next line.
     RETURN                201
```

Listing 3

USR(16495)

```
     LD HL, ADDRESS        33, 128, 64   Point to first character
     LD B, #12             6, 12         of 12 characters.
(1)  LD (HL), #28          54, 28        POKE the first zero.
     INC HL                35            Point to next position.
     DJNZ (1)              16, 251       Repeat 12 times.
     RETURN                201
```

will then exist in memory and will be saved along with the plotting program. After you have run Listing 4, be sure not to key NEW to enter the plotting program—just enter the new lines of code. The line numbers shown in Listing 4 have been chosen to match those of the final program, so you do not have to delete anything.

The purpose of this manner of handling C and D is primarily to conserve memory. The 4K Basic always saves variables along with the program. This is true even if the "variable" is actually an array of constants, or one in which all the elements equal zero. Consequently, it is not always necessary to use up memory with lines of code for creating constants or dimensioning arrays. Nor is it necessary to use cumbersome, slow-running routines to simulate DATA statements. (DATA statements are provided in many version of Basic for just this purpose.) There *are* disadvantages and dangers involved in the technique used here! Again, the intent is not clear unless the documentation spells out what is going on. Also, if the program is kicked off by RUN rather than a GOTO command, the variable storage area of memory will be cleared—*including C and D*! The ZX80 will not recognize C and D when it encounters them in the program; this will produce an error situation and will stop the run.

Listing 4

Loading C and D

```
 10 DIMENSION C(15)
 20 CLS
 30 FOR J=0 TO 15
100 INPUT C(J)
110 PRINT J, C(J), CHR$(C(J))
120 NEXT J
130 DIMENSION D(3)
140 FOR I=0 TO 3
150 PRINT "D(";I;")=";D(I)
160 NEXT I
```

Enter and run this program. Enter, in order; 0, 7, 6, 3, 5, 130, 8, 132, 4, 136, 2, 133, 131, 134, 135, 128. The screen will display the relationship between the binary code (giving the decimal value, J) which the program uses, the ZX80 character code, and the character (group of pixels) being represented. Below that, D(0) through D(3) should show as zero.

From this point on, *do not use RUN or CLEAR* unless you have saved arrays C and D—or you are willing to do this again.

Listing 5

Loading USR(16427)and USR 16495)

```
  1 REM 12345678902234567890323456
 7890423456789052345678906234567 89
 07234
  2 REM 12345678901
 10 LET C=0
 20 FOR I=0 TO 63
 30 INPUT J
100 LET C=C+J
110 PRINT J,
120 POKE 16427+I,J
130 NEXT I
140 PRINT
150 PRINT "CHECKSUM=";C
160 STOP
170 LET C=0
180 FOR I=0 TO 10
200 INPUT J
210 LET C=C+J
220 PRINT J,
230 POKE 16495+I,J
240 NEXT I
250 PRINT
260 PRINT "CHECKSUM=";C
270 PRINT
300 PRINT
310 PRINT
```

Line 1: 64 characters after REM—placeholders for Z80 op codes. If the OR routine is being used, line 2 must have 13 more characters.

Enter and run the program. At the prompts for input, enter the op codes shown in Listing 2 in the sequence shown. Use the numbers in parentheses only if you are creating a 20 by 32 field. After 64 op codes have been entered, execution will stop. The screen should look like this:

```
42      12      64      35
6       4       197     205
94      64      193     16
249     6       7       54
3       35      16      251
62      117     60      119
35      6       5       197
205     94      64      193
16      249     34      14
64      34      16      64
62      14      50      37
64      62      0       50
36      64      201     6
7       54      0       35
16      251     62      117
60      119     35      201

CHECKSUM=4804
```

Compare this to the screen and note any corrections which will be required. Command CONTINUE. Enter the op codes from Listing 3. After 11 op codes have been entered, execution will stop. The screen should look like:

```
33      128     64      6
12      54      28      35
16      251     201

CHECKSUM=828
```

SAVE as soon as you have gotten this far. Remember not to use LIST. LIST 10 (or higher) is fine.

Loading Machine Language Routines

Listing 5 is a program to load the routines shown in Listings 2 and 3. Space is reserved in the REMARK lines 1 and 2 for storing the routines. Enter the program as shown in Listing 5. You may do this either before or after you run the program of Listing 4; if you do it after, remember not to use RUN, but rather GOTO 10. Again, the line numbers have been chosen to avoid the need to delete lines when entering Listing 1.

The PRINT commands which have no obvious purpose are there in order to move lines 1 and 2 off the top of the screen. This is because after the original numbers in those lines have been replaced with Z80 machine language, attempts by the Basic interpreter to produce characters from that language can be fatal. Consequently, after the program has been started, no attempt should be made to LIST the whole program, or lines 1 and 2. LIST 10 is fine. On the other hand, *after* you have saved the routines, go ahead and LIST, just to see the results.

When you run Listing 5, it will stop for the input of the machine language. This is done by providing, in sequential order, the numbers shown in the middle columns of Listings 2 and 3. These numbers are the decimal equivalents of the "op codes"—the numbers which the Z80 will interpret as commands. Each number is stored as one byte in the memory of the ZX80, waiting to be read and interpreted for direct execution by the Z80, bypassing the ZX80 4K Basic interpreter.

As the program proceeds, it accumulates the sum of the op codes which you enter. After you have entered all the op codes for a particular routine, the screen will display all the codes which you have entered, along with the value of that sum—the "checksum." The checksum should be compared with the one shown, providing a fair check for the accuracy of your entries. You should also compare the screen against the copy shown. Corrections can be made by rerunning the entire program, or by POKEing the individual codes. For example, if the fifth code of Listing 1 has been entered as 7, it can be corrected with POKE 16427+4,6.

Notes and Exercises

If you have more than 1K of RAM, you can expand the plotting field. A good size is 20 lines by 32 columns, giving a 40 by 64 pixel field. The changes required in the machine language routines are shown in parentheses in Listing 2. Enter those numbers in place of the ones shown. The checksum will, of course, change accordingly. The corresponding changes which need to be made in the Basic program are:

```
300 FOR I=0 TO 63
340 IF Y<-20 OR Y>18 THEN NEXT I
440 LET Y=(19-Y)/2
450 LET C=PEEK(256*PEEK(16397)+
    PEEK(16396)+33*Y+X)
630 POKE 256*PEEK(16397)+PEEK(1
    6396)+33*Y+X,C(J)
```

Listing 6

USR(16508) and Revised lines 540-560

```
NOP                 0               Used to store J.
NOP                 0               Used to store 2**P.
LD HL, (16506)      42, 122, 64     Put J in register L.
LD A, (16507)       58, 123, 64     Put 2**P in Accumulator.
LD H, #0            38, 0           Clear H (but not L).
OR L                181             OR A with L.
LD L, A             111             Put OR(A,L) in HL.
RETURN              201
```

(Note that 16508 is the address of the third op code in this routine, rather than the first.) The checksum is 1004.)

Revised Basic lines:

```
540 POKE 16506, J
550 POKE 16507,2**J
560 POKE 256*PEEK(16397)+PEEK(16396)+8*Y+X,
    C(USR(16508))
```

(Delete lines 570 through 630.)

Z80 TUITION
USING A 1K ZX81 FOR MACHINE CODE PROGRAMMING

8 Part fortnightly postal course also includes details for adding RAM EPROM HEX KEYPAD LCD DISP LCD DISPLAY & PROM PROGRAMMER.

Course price $49.00 U.S.A./Canada
£21.95 U.K.

Or send $3.50/£1.50 for Z80 Instruction Codes and Course Syllabus.

ANDOVER SOFTWARE KITS

15 Winchester Rd.,
Andover.
Hants SP10 2EG.
England.

EMVEE SOFTWARE ZX81/16K PROGRAM

MINI-GOLF

The Program accurately simulates the game of stroke play golf. You play over 9/18 holes against either another Player or the ZX81 itself. Regardless of the selected length of round, holes are randomly generated in sets of 5 x Par 3;2 x Par 4 & 2 x Par 5 per each 9 holes. You and your opponent select a Club for each shot from identical sets of Driver; 1 thro 9 Irons;Pitching Wedge & Putter, each of which has a predetermined "standard" yardage. The better a Player is performing to Par the less the deviation is likely to be between the "standard" yardage for a particular Club and the actual yardage achieved from its use; conversely the worse a Player is performing to Par, the greater the deviation is likely to be. You may also elect to play a "soft" shot thereby reducing the potential distance that will be achieved from the use of a Club. As an overall probability, the better your performance is likely to become as the holes progress.

The screen constantly displays a "score-card" for the current 9 holes being played (with carried-forward totals included for an inward 9 holes), which shows for each hole - the total length; the length to the front edge of the Green; Par for the hole. For holes already completed the display shows for each Player - number of strokes taken; relative performance to Par per hole; cumulative performance to Par so far. For the current hole being played the display shows for each Player - shots taken so far; remaining distance to the Pin (which on reaching the Green is automatically converted from yards to feet). If you have a ZX Printer, the option exists to obtain a print-out of the "score-card" at the end of the outward and/or inward halves.

Because of the cost, and other potential problems associated with posting Program Cassettes to overseas clients, the Program is supplied in the form of a Program Listing (produced by a ZX Printer) from which you merely have to key-in the BASIC instructions. The listing is available together with a fully detailed Instruction Booklet and details of other Program Products, for just $6.00 (inc airmail postage). Interested? then please post the coupon below:

* * * * * * * * * * * *

Please forward a copy of ZX81/16K "Mini-Golf" for which I enclose $6.00 (Please also enclose full postal address details).

EMVEE SOFTWARE,
10 Mythop Road, Lytham FY8 4JD, Lancs, England.

Another useful change allowed by more RAM would be the usage of more lengthy expressions. Determine the number of characters which you wish to permit in the expression; add two to that, and place that many zeroes in line 10. Change the 12 in Listing 2 to that number as well.

You do not have more RAM? There may be some things you can do to stream line the program, even in Basic. Real size advantages can be had in machine language, though. For example, you can make use of the Z80 OR operation. (See *SYNC* 1:6 pp. 9-11). OR compares the contents of the accumulator register, A, with a chosen internal register, and performs a logical OR on the contents, bit by bit. The results are placed in the accumulator. Suppose, for instance, that A contains 6 and register L contains 3:

Register	Decimal	Binary
A	6	0110
L	3	0011
OR(A,L)	7	0111

Actually, each register contains eight bits rather than four. Only four are shown here to help clarify the point: The operations carried out in lines 540-620, using array D, could be replaced with an OR of J and P**2. Of course, that requires another machine language routine, and some Basic to set things up. Both are shown in Listing 6. The machine language routine makes use of a characteristic of the USR function: USR returns a value—whatever number is in the HL pair of internal registers when the return to Basic is made. Consequently, L is chosen as the register to be ORed with A, H is set equal to zero, and the results are copied from A into L before return. Since L is the least significant byte of the HL pair, and H is zero, the value returned to the Basic program is the value of L alone:
OR(P**2,J).
The savings are significant: Lines 540-630 take up 123 bytes of memory; array D takes up another 10 bytes. Replacement lines 540-560 require 75 bytes and the machine language routine only 13 bytes. (Since we are trying now for better efficiency in the use of storage, I am not including bytes for a third REM line.) If I counted everything properly, the savings are 123+10-75-13 = 45 bytes. That is enough for another four columns or six rows!

Not only that, but surely the Basic program can be improved on. If you get too tricky, remember your documentation!

With more RAM and/or ingenuity, you could certainly add features to the screen. How about:

Drawing a reference line for X? (Poke a series of characters in a vertical line at some value of X or I. The value could be preselected or user defined. This line could serve as the Y axis.)

Labeling the axes? (Poke the characters X+ at or by the right end of the X axis and Y+ at or by the top of the Y axis.

Finding and labeling the extreme values? (Create two variables—perhaps YMAX and YMIN—and compare each new value of Y to them. Keep the highest or lowest each time. Record their addresses in ADMAX and ADMIN. If you are using a 32 column screen, the addresses would be 33*Y+X, after you have made the conversion from the Y returned by subroutine 10 to the Y used in the field coordinates. Label by POKEing a special character at those addresses
(+256*PEEK(16397)+PEEK(16396))
and printing the values elsewhere.)

Printing the expression below the field? (Remember that you have a limited number of rows available for printing.)

Clearing only the lower portion of the screen in order to take in different values of XM, XS and YS after the More response? (Perhaps you could have two More options: with the old values and with new values. Then you could easily compare the effects of scale changes. You should be able to adapt portions of Listing 2 to POKE a series of spaces into the lower lines.)

Whatever you do, however much RAM you have, above all, you should enjoy yourseelf while using this program. For me, that is the major point of entire thing.

Two last pieces of advice:

If the screen goes bonkers or the program crashes, suspect the machine language first. Take a PEEK at all such routines and compare them carefully to the listings. Make sure that the addresses in your USR calls are correct. Confirm that the byte located at each of those addresses is the one that should be there. However, if the program runs, but plots the wrong pixels, check the lines that PEEK at and POKE to screen addresses.

Finally, when the program runs smoothly in its original format, save it and keep it handy to show to friends who wonder what your ZX80 can do besides games. Make another copy to experiment with—leave the original alone until you have another version that works as well.

Have fun! ■

TIMES SQUARE ON YOUR ZX81

LSCROLL—For Spectacular Screen Displays

Douglass D. Sharp

How would you like to be able to create a "Times Square" display which moves from right to left without upsetting other items on your ZX81 (or 8K ROM ZX80) screen? Would you like to display graphs or functions such as a sine wave on your screen when they need more than 32 columns of display? Or what about moving animated characters or graphs across the screen with great speed? You can easily do these and more with LSCROLL, a lateral, or horizontal, scrolling routine for your computer.

Moving graphics around is not extremely difficult with the 8K ROM by using the SCROLL, PAUSE, PRINT AT, PRINT TAB, and UNPLOT statements. Speed becomes important, though, even in the FAST mode. The program described here, written especially for 16K RAM (because of machine language coding), adds to the graphics capabilities of your computer by allowing pictures to move horizontally across the screen without changing their shapes.

Lateral SCROLLing may be obtained by writing a Basic program which uses the string handling capabilities of the 8K ROM. Again, speed is important. Owners of the ZX80 would be staring at a blank screen during execution of the program, and the ZX81 users would wait even longer if the SLOW mode is used. LSCROLL solves these problems. Execution of the lateral scrolling program shown here is extremely fast—under certain conditions even faster than the SCROLL statement.

Here are some of LSCROLL's features:

1) Machine Language written—fast in both the SLOW and FAST modes and protected by the RAMTOP.

2) 172 bytes long.

3) Beginning line to scroll is selectable by the programmer.

4) Range of lines to be scrolled is selectable by the programmer.

5) Fill characters to be placed in newly opened columns are selectable by the programmer.

6) Totally dynamic—works even if SCROLL is used in your program.

7) 22 lines scrolled laterally, maximum.

8) Scrolling is from right to left—ideal for text windows.

Before writing LSCROLL, I needed to understand the format of the display file within the computer. It is comprised of 24 lines of text, each beginning with an ENTER character, code 118. Some trailing blanks of lines of text are eliminated to save space in the RAM under certain conditions during program execution of SCROLL, or even when typing is being performed such as when a program is being written into the computer. The LSCROLL program takes these conditions into account to avoid crashing the system by upsetting the display file.

Figure 1: LSCROLL Listing.

```
   4 LET C=0
   5 REM ALL OS IN THIS LISTING
   6 REM ARE ZEROS.
  10 LET D$="3AEE7FFE16DA517F3E0
032EE7F3AEF7FD600CA617FFE15D2617
FC3667F3E1632EF7FF53AEF7F47F13AE
E7F80FE16DA857F2600?E163AEE7F4F0
600ED427D32EF7FED5B0C400EFFF53AE
E7F6FF1260006001AFE76CA9F7F13C39
57F0CA7E5ED42E1CAAC7F13C3957F060.
0F53AEF7F4FF1C5D5E1237EFE76C2B77
FE5A7ED522B7D4DD600E1C2D07FE5D1C
3E57F3DC2D87F13C3E17F06000B13D5E
123EDB03E00121 3C10B79D600C2B47FC
90000"
  20 FOR X=2 TO 344 STEP 2
  30 LET P=(CODE D$(X-1)-28)*16
  40 LET P=P+CODE D$(X)-28
  50 POKE 32579+INT (X/2),P
  60 LET C=C+P
  70 NEXT X
  80 PRINT C
```

To use LSCROLL, shown in Figure 1, do the following:

1) *Carefully* type in the program. You may want to execute a line 10 as a command—EDIT, DELETE, DELETE, ENTER, and then PRINT LEN D$. The value printed after this should be 344.

2) SAVE your proofread program on tape.

3) Enter NEW.

4) Enter POKE 16388,68 and POKE 16389,127. Enter NEW. This will protect LSCROLL from being written over by the Basic system.

5) LOAD your saved program, and RUN it. If within several seconds, the number 22093 appears, enter NEW. If this did not happen, go back to step 1.

Other ways of loading LSCROLL into your computer will be discussed later. For now, since you have LSCROLL in your computer, let us do some experimenting.

To begin scrolling the screen at line XX (0 is the top line, 21 is the last line you can PRINT) just POKE 32750,XX. To scroll for YY lines after and including line number XX, POKE 32751,YY. To execute scrolling, try RAND USR 32580.

As an example, let's begin with a problem which requires scrolling the third line of the screen. We might also wish to scroll the remainder of the screen. The top two

Figure 2: Illustrative Use of LSCROLL.

```
  10 PRINT AT 0,13;"TEST";AT 5,1
3;"HELLO";AT 20,13;"GOODBYE"
  20 POKE 32750,2
  30 POKE 32751,20
  40 PAUSE 20
  50 RAND USR 32580
  60 GOTO 40
```

Douglass D. Sharp, 10914 Beacon St., Kansas City, MO 64134.

lines of the screen, numbered 0 and 1, are not to be changed (note that LSCROLL uses the same line numbering system as Sinclair). Enter the program shown in Figure 2 and RUN it. You will want to BREAK after scrolling is complete.

Note that each execution of line 50 in the program will scroll from right to left. Of course, you want to scroll only a certain number of columns. There are several ways to do this. One way is to set up a FOR/NEXT loop around lines 40 and 50. Another way is to change line 50 to something like 50 LET X=X+1+USR 32580. The USR routine always returns the value zero, so the effect of this statement is to increment X for every scroll. The value of X can then be tested with an IF statement for appropriate action.

Also notice that blanks are "filled" into newly opened columns on the screen from the right as characters are LSCROLLed to the left. To "fill" with a different character, just type POKE 32738,CC where CC is the code of the character to be used for filling. For example, to fill with an inverse video blank, enter POKE 32738,128 or even POKE 32738,CODE "■".

There is another thing interesting about LSCROLL. Look at Figure 2. Values of XX, YY and CC mentioned above need not be POKEd each time LSCROLL is used. When any changes are to be made to these values, only the individual POKE statement needs to be used first. If XX and YY are never POKEd, LSCROLL assumes that the entire screen is to be LSCROLLed. The chart in Figure 3 shows values of XX, YY and CC, along with their default (assumed by the program) values.

LSCROLL is executed by using the sequence of characters "USR 32580" in certain places in your program. The RAND statement, as shown in line 50 of Figure 2, can be used. This makes LSCROLL very convenient to call only when random numbers are used, USR 32580 serves as an "argument" in a dummy LET statement such as LET ZZ=USR 32580.

Figure 4 shows several applications of LSCROLL. Try putting 95 SCROLL into the first program and see how that runs. With this, it looks as though your ZX80 or ZX81 could cost ten times what it does. Try running all of these programs. This may give you some ideas for more applications of LSCROLL. Be sure to NEW the computer between each of the programs. Do not worry. RAMTOP was modified and LSCROLL is protected. The line pointer and range remain as last POKEd, so each program shown ensures that they are selected and POKEd properly. If you ever wonder what values are POKEd in, just PEEK them out with commands. Remember, though, that LSCROLL may have modified any invalid numbers you POKEd, according to the rules in Figure 3.

Figure 3: Values of XX, YY, and CC.

VALUE	POKE ADDRESS	DEFAULTED VALUE	FUNCTION
XX	32750	0 if XX> 21	Specifies the line number to begin scrolling. (0-21)
YY	32751	22 (all lines) if XX+YY> 22 or if YY=0	Specifies the number of lines after and including line XX to be scrolled. (1-22)
CC	32738	0 if not POKEd but will not be altered or checked	Specifies the CODE of the character for "filling" in newly scrolled columns.

To execute LSCROLL, use USR(32580) in your program.

To load LSCROLL, NEW/ POKE 16388,68/POKE 16389,127/NEW/ read LSCROLL in/RUN

Figure 4: Applications of LSCROLL.

\# indicates a blank

```
 10 REM PROGRAM 1
 11 REM FOR ZX81 SLOW MODE,
 12 REM REMOVE LINE 70.
 20 POKE 32750,10
 30 POKE 32751,1
 40 LET S$="TIMES SQUARE DISPLA
Y#####"
 50 FOR X=1 TO 25
 60 PRINT AT 10,31;S$(X)
 70 PAUSE 10
 80 RAND USR 32580
 90 NEXT X
100 GOTO 50
```

```
 10 REM PROGRAM 2
 20 PRINT AT 5,0;
 30 LIST
 40 POKE 32750,5
 50 POKE 32751,100
 60 PAUSE 10
 70 RAND USR 32580
 80 GOTO 60
```

```
 10 REM PROGRAM 3
 20 LET S$="##NUMBER#"
 30 POKE 32750,10
 40 POKE 32751,1
 50 FOR X=1 TO 20
 60 LET D$=S$+STR$ X
 70 FOR I=1 TO LEN D$
 80 PRINT AT 10,10;"#";AT 10,20
;D$(I)
 90 PAUSE 10
100 RAND USR 32580
110 NEXT I
120 NEXT X
```

```
 10 REM PROGRAM 4
 20 POKE 32750,0
 30 POKE 32751,22
 40 PLOT RND*63,RND*43
 50 PAUSE 10
 60 LET X=USR 32580
 70 GOTO 40
```

```
 10 REM PROGRAM 5
 20 POKE 32750,0
 30 POKE 32751,50
 40 POKE 32738,8
 50 FOR Y=21  TO RND*21 STEP -1
 60 PRINT AT Y,31;"■"
 70 NEXT Y
 80 LET X=USR 32580
 90 PAUSE 10
100 GOTO 40
```

```
 10 REM PROGRAM 6
 11 REM MODIFIED FROM THE ZX81
 12 REM BASIC MANUAL,
 13 REM SINCLAIR RESEARCH, LTD
 20 POKE 32750,0
 30 POKE 32751,0
 40 POKE 32738,0
 50 FOR X=1 TO 200
 60 PLOT 63,22+20*SIN (X/32*PI)
 70 PLOT 63,22+20*COS (X/32*PI)
 80 PAUSE 8
 90 RAND USR 32580
100 NEXT X
```

How does LSCROLL work? Figure 5 shows the standard Z80 source listing of LSCROLL. Since the text of the display file may change in length and format under certain conditions, LSCROLL takes great care in checking the status of the display file's area in RAM.

The LSCROLL program begins by checking both the line pointer and the range for scrolling as to the validity of these values. Adjustments are made according to the rules in Figure 3 to insure that LSCROLL does not crash the system. The fill character is not checked.

Each line of text in the display file begins with an ENTER character. The LSCROLL program scans the file and counts ENTER characters until the first line to be scrolled according to the POKEd line pointer is found.

Figure 5: Z80 Source Listing of LSCROLL.

Address	Object Code	Ref	Instruction		Comments
32580	3AEE7F		LD	A,(LBL21)	;get line pointer in acc.
32583	FE16		CP	22	;compare with 21.
32585	DA517F		JP	C,LBL1	;≤21 go to lbl1.
32588	3E00		LD	A,0	;clear acc.
32590	32EE7F		LD	(LBL21),A	;make line pointer zero.
32593 7F51	3AEF7F	LBL1	LD	A,(LBL22)	;get range in acc.
32596	D600		SUB	A,0	;test acc.
32598	CA617F		JP	Z,LBL2	;range=0? yes-go to lbl2.
32601	FE15		CP	21	;compare with 22.
32603	D2617F		JP	NC,LBL2	;>21 go to lbl2.
32606	C3667F		JP	LBL3	;go to lbl2.
32609 7F61	3E16	LBL2	LD	A,22	;load acc with 22.
32611	32EF7F		LD	(LBL22),A	;make range=22.
32614	F5		PUSH	AF	;load range in
32615 7F66	3AEF7F	LBL3	LD	A,(LBL22)	;B and line
32618	47		LD	B,A	;pointer in acc.
32619	F1		POP	AF	;for final validity
32620	3AEE7F		LD	A,(LBL21)	;comparison.
32623	80		ADD	A,B	;add the two.
32624	FE16		CP	22	;compare the sum with 22.
32626	DA857F		JP	C,LBL4	;sum ≤ 22 go to lbl4.
32629	2600		LD	H,0	;otherwise,
32631	2E16		LD	L,22	;make the
32633	3AEE7F		LD	A,(LBL21)	;range
32636	4F		LD	C,A	;equal to
32637	0600		LD	B,0	;the difference
32639	ED42		SBC	HL,BC	;(22-line pointer)
32641	7D		LD	A,L	;to insure the

EZRA GROUP II
EZRA GROUP II

The ZX81/80's are making a name
with LOW prices...

WE CHALLENGE THE SOFTWARE COMPANIES TO LOWER THEIR PRICES!

for ZX81/ZX80/8K ROM
1K and 16K RAM versions

Biorhythms	1.00
Graphics Billboard	1.00
Horse Race	1.00
SPINNER T.M. (like Rubik's) 16K	2.00
Skew-a-Sketch (like Etch)	1.00
Improved Pause (ZX81)	1.00
Linear Regression	2.00
Linear Programming	Ask.
Shootist	2.00

Self Addressed Stamped Envelope
Gets YOU our Goodies Catalog

ALL ORDERS AND CATALOG REQUESTS GET **FREE** Galactic Messages PROGRAM.

EZRA GROUP II
EZRA GROUP II

P.O. Box 5222 San Diego, California 92105 (714) 584-8291

INCREDIBLE!

ZX80/1 Users

RKL-16 Expanded Memory - 16K	79.95
RKL-32 Expanded Memory - 32K	149.95
RKL-J1' Single Joystick System	69.95
RKL-J2' Dual Joystick System	89.95

30-Day Money Back Guarantee
Full Assembled and Tested
**Includes Demonstration Program*

SUPER PACKAGES!

Super RKL-A *32K Memory plus Single Joystick System*	199.95
Super RKL-B *32K Memory plus Dual Joystick System*	219.95

RKL Systems

Include 4.95/shipping + handling
Check or money order

PO Box 515
Leominster, MA 01453

Figure 5: continued

Address	Object code	Ref	Instruction	Comments
32642	32EF7F		LD (LBL22),A	;correct end of scroll.
32645 7F85	ED5B0C40	LBL4	LD DE,(16396)	;load D_FILE in DE pair.
32649	0EFF		LD C,255	;set counter to -1.
32651	F5		PUSH AF	;load the
32652	3AEE7F		LD A,(LBL21)	;line pointer
32655	6F		LD L,A	;into
32656	F1		POP AF	;the HL
32657	2600		LD H,0	;register pair.
32659	0600		LD B,0	;make BC pair a counter.
32661 7FA5	1A	LBL5	LD A,(DE)	;compair display file
32662	FE76		CP 118	;character with 118.
32664	CA9F7F		JP Z,LBL6	;not 118(ENTER) go to lbl6.
32667	13		INC DE	;inc. search address.
32668	C3957F		JP LBL5	;loop back to lbl5.
32671 7F9F	0C	LBL6	INC C	;ENTER found, inc. counter.
32672	A7		AND A	;clear carry flag.
32673	E5		PUSH HL	;subtract counter
32674	ED42		SBC HL,BC	;from
32676	E1		POP HL	;line pointer.
32677	CAAC7F		JP Z,LBL11	;result=0, go to lbl11.
32680	13		INC DE	;inc. search address.
32681	C3957F		JP LBL5	;loop back to lbl5.
32684 7FAC	0600	LBL11	LD B,0	;load BC
32686	F5		PUSH AF	;register
32687	3AEF7F		LD A,(LBL22)	;pair
32690	4F		LD C,A	;with
32691	F1		POP AF	;range.
32692 7FC4	C5	LBL12	PUSH BC	;save BC pair in stack.
32693	D5		PUSH DE	;put addr. of first line
32694	E1		POP HL	;to scroll in HL pair.
32695 7FB7	23	LBL13	INC HL	;point to next char.
32696	7E		LD A,(HL)	;get char. in acc.
32697	FE76		CP 118	;compare with 118.
32699	C2B77F		JP NZ,LBL13	;not ENTER go to lbl13.
32702	E5		PUSH HL	;save end of line in stack.
32703	A7		AND A	;clear carry flag.
32704	ED52		SBC HL,DE	;put length of line in HL
32706	2B		DEC HL	;pair less one for ENTER.
32707	7D		LD A,L	;put length in acc.
32708	4D		LD C,L	;also save length in BC pair.
32709	D600		SUB A,0	;test acc.
32711	E1		POP HL	;get line addr. back.
32712	C2D07F		JP NZ,LBL15	;acc not zero go to lbl15
32715	E5		PUSH HL	;put address of line in
32716	D1		POP DE	;DE pair (no scroll)
32717	C3E57F		JP LBL23	;go to lbl23
32720 7FD0	3D	LBL15	DEC A	;decrease acc. if acc.
32721	C2D87F		JP NZ,LBL18	;was < 1 go to lbl18.
32724	13		INC DE	;increment pointer, then
32725	C3E17F		JP LBL19	;go to lbl19 for fill.
32728 7FD8	0600	LBL18	LD B,0	;clear B.
32730	0B		DEC BC	;decrement counter.
32731	13		INC DE	;set up scroll pointer.
32732	D5		PUSH DE	;place pointer in
32733	E1		POP HL	;HL pair and increment
32734	23		INC HL	;for fetching address.
32735	EDB0		LDIR	;scroll the line
32737 7FE1	3E00	LBL19	LD A,0	;fill character into acc.
32739	12		LD (DE),A	;acc. into last char. pos.
32740 7FE5	13	LBL23	INC DE	;point to next line.
32741	C1		POP BC	;decrement range counter
32742	0B		DEC BC	;to ready next scroll.
32743	79		LD A,C	;load acc. with count.
32744	D600		SUB A,0	;test acc. range not
32746	C2B47F		JP NZ,LBL12	;statisfied, go to lbl12
32749 7FED	C9	LBL20	RET	;return to BASIC BC is 0.
32750 7FEE	00	LBL21	NOP	;storage for line pointer.
32751 7FEF	00	LBL22	NOP	;storage for range.

Before any line is scrolled to the left, LSCROLL counts the length of that line by looking for the next ENTER character. If the length is less than one character, no scrolling takes place.

LSCROLL scrolls until all lines specified via the range value have been shifted. Actual shifting is performed by the Z80 LDIR, "Repeating Block Load with Increment," instruction. The contents of the memory location whose address is in the computer's HL register pair are loaded into the memory location whose address is in the computer's DE register pair. Then, DE and HL are both incremented by one, and BC is decremented by one. When BC, holding the length of the line of text to be shifted, is zero. LDIR allows the computer to execute the next instruction.

You can use LSCROLL in your programs either by entering LSCROLL and NEWing it, as described above, or by writing a program around the listing in Figure 1. Of course, LSCROLL should be on tape to save a lot of typing.

If a program is written around the listing, all that must be done before you LOAD it is the two POKE commands that modify RAMTOP described earlier. It would be a good idea to put the statement:

```
XXXX IF PEEK 16388+PEEK 16389
*256 <>32580 THEN STOP
```

somewhere near the beginning. This will make sure you did your POKE commands before execution continues. Do not forget that your program cannot do these POKEs because NEW must be executed immediately afterwards. You could spell out these two POKE commands in a REM statement for reference later.

Use your imagination for finding uses for LSCROLL. How about bar charts more than 32 characters wide? Or even objects racing across the screen in games? Try laterally scrolling prompts to users before INPUT statements to break monotony. Also think about this: When LSCROLL and SCROLL are used alternately, an effective diagonal movement of screen graphics or characters results. Virtually every program imaginable could benefit from the frills of the 172-byte LSCROLL program.

Part 2

How to Invent a Game

Jonathan Bobst

In Part 1, we covered the conception and initial game design of *Flattop Lander*, and we examined the Machine Code subroutine that will produce an instant display.

To finish up the display, let's put the carrier characters into the storage line (2 REM) using the MC-input routine:

```
100 FOR X=16486 TO 16493
110 INPUT Y
120 POKE X,Y
130 PRINT PEEK(X),
140 NEXT X
```

GO TO 100 and enter these 8 character codes: 134,128,128,128,128,128,128, and 135. At this point, the display is complete with aircraft carrier and ocean surface, so, if you have entered lines 1, 2, and 40, and lines 90-180, you can GOTO 40 to check it.

The variable "C"(Change in direction) was initialized at "9" in line 40 and is added to "P"(Position/address of display in RAM) to put the aircraft in its initial position in the upper left corner of the display "block." Now, we need an input routine to enable the player to move the aircraft in any forward direction.

In Part 1, the "3D" input matrix was explained, so that 7 means "straight ahead—no turn," etc. See Figure 1.

Since the display is 32 spaces wide, dropping one row means having to add 33 to the last aircraft position/address("C"), while rising one row means "C" must be decremented by 33. Taking that "vertical" (and "L" for horizontal) movement into account, we have the finished input routine shown in Figure 2.

Jonathan Bobst, ZETA Software, P.O. Box 3522, Greenville, SC 29608.

So far, the game is not particularly challenging...input a move and the aircraft moves. Big deal. What all games need is an element of chance, so let us add some computer-generated "Lady Luck" to simulate wind or ship movements. See Figure 3.

Next we need some test lines for display output to simulate the deck flagman or instruments. See Figure 4.

Okay, now we have a loop that accomplishes both input and display. Lines 250-

Figure 1.

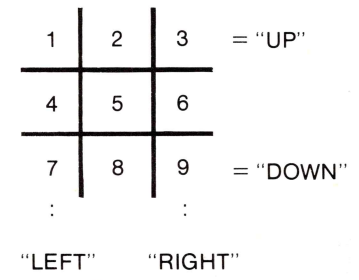


Figure 2.

```
250 INPUT M                              :Enter 0-9.
260 IF M=0 THEN GOTO 30                  :"Reset" for a new approach.
270 IF M>6 THEN LET C=C+33               :Drop down 1 row.
280 IF M<4 THEN LET C=C-33               :Rise up 1 row.
290 IF M=3 OR M=6 OR M=9 THEN L          :To move right to get on approach
    ET L=L+1                              line.
300 IF M=1 OR M=4 OR M=7 THEN L          :To move left.
    ET L=L-1
310 LET C=C+1                            :This aircraft flies forward only.
320 LET D=D-50                           :Distance decrementer.
330 LET H=400-((C-9)/33)*100             :Height decrementer.
340 GOTO 80                              :For the game's ""loop."
```

Figure 3.

```
30 RANDOMISE
70 LET L=0                               :Initialize "L" (approach line).
80 LET L=L+RND(3)-2                      :"L" plus -1, 0, or +1.
```

Figure 4.

```
190 PRINT H,D                            :Show height and distance to
                                          catchcable.
200 IF L<0 THEN PRINT ">"                :"Move right to get on approach
                                          line."
210 IF L>0 THEN PRINT "<"                :"Move left."
220 IF L=0 THEN PRINT "X"                :"On line."
```

340 recalculate the variables, and lines 70-220 produce the display from those variables. But, it is still not a true game without a scoring mechanism or at least a win/lose test. By limiting the display area in the design stage, we have enough RAM left to include both. See Figure 5.

After winning another point or losing everything, a player needs to be asked if he wants to continue, hence, lines 520-540. They are terse because, at this point, the program is very close to filling up the 1024 bytes in 1K RAM. To make this "endgame" routine even shorter, line 530 can be omitted, but the player must then enter an unused variable name to make the program stop (on error code 2).

After hours of thinking and planning and typing-in statements, the program is finally complete...or is it? If run as is, it will stop on error code 2/190 even before asking for the first move. By checking line 190 (PRINT H,D), it is apparent that we have not *named* either variable, yet. So:

50 LET H=400
60 LET D=1000

Line 50 shows the height above deck, and 60, the distance to the catchcable on deck.

Now the program will run as planned *and* within 1K of RAM.

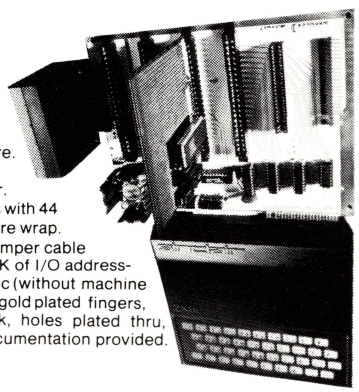
Inventing a computer game is a *reciprocal*, creative process—while you are developing the game from first idea to finished program, the game is developing *you*. ■

Figure 5.

```
 20 LET S=0                          :Initialize Score.
230 IF D=0 AND H=0 AND L=0 THEN      :Test for safe landing to exit
    GOTO 500                          loop.
240 IF C>161 THEN GOTO 400           :Test for a crash.
400 PRINT "*CRASH*"
410 POKE P+C,59                      :Show explosion CHR$(59).
420 LET S=0                          :Take away player's points for
                                      crashing.
430 GOTO 510                         :Skip over "win."
500 LET S=S+1                        :Increment score.
510 PRINT "SCORE=";S
520 INPUT X                          :Enter any number to continue or
530 IF X=0 THEN STOP                 :"0" to STOP.
540 GOTO 30                          :"Reset" to initial aircraft
                                      position.
```


EXPANSION BOARD FOR SINCLAIR (tm) COMPUTERS

ZX80*/81/Microace*
Buffered bus for memory mapped peripherals



Offers significant yet inexpensive access to ZX80/81 hardware. Interfaces existing hardware. Allows users to develop their own hardware. Includes all the logic that appears on Sinclair's port in the same order. Accommodates popular circuit cards with 44 fingers at .156 in. spacing. Room for wire wrap. Also accommodates connector and jumper cable hardware with .1 in. spacing. Up to 8K of I/O address-ability can be programmed via Basic (without machine language). Provision for 3 A regulator, gold plated fingers, fastens to ZX81, legend, solder mask, holes plated thru, many on board options. Complete documentation provided.

BARE BOARD with the computer connector is:
$34 postpaid. (all parts readily available)

Product price/availability subject to change. Allow up to 6 weeks for delivery. *(ZX80 will require a minor circuit modification.)

To order send check to:

Computer Continuum

301-16 ave. San Francisco, Ca. 94118
(415) 752-6294 **Feb. 82**

THE EXPLORER'S GUIDE TO THE ZX81

If you have a ZX81
then you *need* this book.
1K and 16K programs.
Games and Applications.
RAM and I/O circuits.
Programming hints.
ROM routines.
$10 from:
TIMEDATA
3 Waldon Road,
Califon, NJ 07830

Safe Machine Code Routines

Harry Doakes

More and more programs are appearing in *SYNC.* under the rubric of "machine language." There was a time, not too long ago, when personal computer users were notoriously shy about venturing into these jungles. Nowadays, though, even someone who is not quite comfortable in Basic can manage to key in and run programs in Z80 machine code.

Unfortunately, there are still a few bugs in the system. The Sinclair machines are really designed for Basic, not for special machine code routines. True, there are commands—PEEK, POKE, and USR—that are handy for putting these fast, specialized routines wherever the programmer desires.

Unfortunately, there is nowhere to put them.

"This routine is held in a REM statement that must be kept off-screen. Don't try to LIST the program," says one author. Says another, "I've included extra code to prevent the system from bombing if you try to LIST it."

Some programmers like to put their machine code routines in the space reserved for variables—but, as one writer points out, "I haven't yet figured out how to keep from losing the routine when you RUN the program."

What's the problem, anyway? Why can't these programmers find a safe, sensible place to put their clever little special routines?

Why use Z80 machine code, anyway? Because it is very compact and very fast. The equivalent of

```
LET A=A+1
```

in machine code takes just one byte, and the Z80 can execute that command in a little more than a millionth of a second. For speed, Basic cannot come close.

Basic is a "high level" language. A program in Basic reads a lot like English, and it can be moved rather easily from one computer to another. But computers do not really understand Basic. To the ZX80, the line

```
10 LET A=A+1
```

is just a string of numbers:
0 10 248 38 227 38 221 29 118.

The only thing the ZX80's microprocessor really understands is its own language. The 4K or 8K Basic ROM contains a program that enables the computer to figure out what to do for each Basic command. (That program, of course, is written in machine language.)

In other words, Basic takes the computer extra work—and extra time.

Machine code programs speak the computer's own language. They are strings of numbers, too—but the processor can understand each number on its own, without a special translator or interpreter program. Unlike Sinclair Basic, Z80 machine language uses *all* of the numbers it can, from 0 to 255—every different value a byte of memory can hold.

That is where the problems begin.

The ZX80 is a *very* small system. It is cleverly designed, and for the small number of components on its printed circuit board, it is amazing for what it does.

But it all depends on several assumptions. For example, when it is time to refresh the TV screen, the ZX80 zips through display memory at incredible speed to throw the proper information into the proper order fast enough to keep up with the TV's electron gun.

To work that fast, the ZX80 was designed so that only regular or reversed characters would be stored in the display section of memory. (Remember, regular characters are represented by numbers from 0 to 63, and reversed characters run from 128 to 191.) The one exception is the NEWLINE code, which is 118.

What happens when a number between 64 and 127, or higher than 192, is in display memory? It is easy enough to see. Enter this line:

```
10 REM THIS IS HOW IT LOOKS
```

Now POKE one of the "forbidden" numbers into 16440, and watch what happens.

Not all of the numbers have the same effect. Sometimes the display rolls; sometimes it is warped; sometimes it just gets a bit jittery. Almost always, given enough time, the whole system will crash.

That is why some authors tell you not to LIST the REM line that contains a machine code routine. If the routine contains any of the "forbidden" numbers—half of all the available instructions—you will probably crash the system and have to start over from the beginning.

There are other assumptions built into the ZX80 system, too. For example, when you RUN a program, the ZX80 assumes that you will want all the variables cleared out, and that you will want to have as much space as possible for your Basic

Harry Doakes, P.O. Box 10860, Chicago, IL 60610.

Listing 1.

```
      1 REM 21.00.00.39.ED.5B.10.40.ED.52.C9
      2 DIM Z(5)
      3 LET Z=USR(48)
      4 FOR X=1 TO 11
      5 LET Z=Z-1
      6 POKE Z, PEEK(16428+3*11-3*X)-28
+16*(PEEK(16427+3*11-3*X)-28)
      7 NEXT X
     10 PRINT "FREE MEMORY=";USR(Z);
" BYTES"
```

Listing 2.

```
21 00 00        LD HL, 0000
39              ADD HL, SP        get stack pointer
ED 5B 10 40     LD DE, (4010)     get end of display
ED 52           SBC HL, DE        subtract
C9              RET
```

program and display. That means memory is used very efficiently in the ZX80.

It also means that most machine language programs cannot be safely stored in REM lines, strings, or the like. It is tough to find a safe place to put a machine code routine.

How can it be done? Listing 1 demonstrates one way. This method takes up program space, but it has three very large advantages. First, it puts the routine in every time you run the program. That means you do not have to remember to replace RUN with GOTO.

Second, you do not have to remember a five-digit number to use the routine. Instead of something like USR(17694), simply use USR (Z).

Finally, you can check, edit, or LIST the routine any time you like, in hexadecimal notation—the form in which most machine code programs are written. It is easier to read, easier to change, and much less prone to mistakes.

Enter and run the program in Listing 1. The simple machine code routine in the REM line calculates the amount of free memory left in a ZX80 system. It works for any size of RAM, but it is designed for 4K Basic, not the 8K ROM.

To put the machine code routine into memory properly, line 2 sets up an array, and line 3 finds the end of the array. USR(48) always returns the location of the end of the storage space for Basic variables. Then the FOR...NEXT loop takes each pair of characters in the REM line, converts it to a single value betwen 0 and 255, and POKEs it into a byte in the array. When the loop is finished, Z points to the start of the routine in memory.

Line 10 simply demonstrates the routine.

Now add these lines and run the program again:

```
20 FOR X=0 TO 5
30 PRINT "Z(";X;")=",Z(X)
40 NEXT X
```

What you get should look something like Figure A.

Figure A.

```
FREE MEMORY=728 BYTES
Z(0)=    8448
Z(1)=    0
Z(2)=    -4807
Z(3)=    4187
Z(4)=    -4800
Z(5)=    -13998
```

It is the machine code routine again—this time as it is stored in the Z array.

Listing 2 shows the routine in the traditional "assembly language" notation.

Many of the machine-language routines printed in *SYNC* can be adapted directly to this method. Simply copy the two-digit hexadecimal numbers into the REM line and adjust lines 2, 5, and 6.

In line 2, DIM the array for half the number of bytes in the routine. For example, a routine 77 bytes long should use DIM Z(38).

Then line 5 would be

```
FOR X=1 TO 77.
```

And line 6:

```
POKE Z, PEEK(16428+3*77-3*X)-28+
  16*(PEEK(16427+3*77-3*X)-28).
```

Line 6 is in expanded form here simply to make it clear that 77 must be inserted in the line twice—once for each of the PEEKs. It could be simplified to:

```
6 POKE Z, PEEK(16659-3*X)+16*PEEK
  (16658-3*X)-476
```

and work just as well.

There are a few things to be careful about.

Be sure to put the periods into the REM line. You can replace them with commas or spaces if you like, but be sure to put something there: they equally space the bytes, and allow the program to put the machine code routine together properly.

If you have a longer routine and want to save space, you *can* leave out the periods—but you will have to change line 6. In the 77-byte example, it would now read:

```
6 POKE Z, PEEK(16428+2*77-2*X)-28+
  16*(PEEK(16427+2*77-2*X)-28)
```

Notice that a "3" has been changed to a "2" in four places—twice for each PEEK.

Be sure to put this routine at the beginning of your program—before you use any of the variables. If you like, you can choose a different variable and array. But *do not* use the array or variable for anything else in your program, or you may crash the system.

Finally, if you are not sure whether a particular machine language routine will work with this system, the easiest way is to *try it*! Instead of the tedious system of keying in two- and three-digit numbers, where one slip could be fatal, simply SAVE the program and change the REM line to install the new routine.

It is an easy way to experiment with machine language programs—and to get more out of your ZX80.

try this

This columns will feature short programs to show off your computer, impress your family and friends, and tickle your imagination when *SYNC* arrives at your place. We invite your contributions. Address them to: Try This, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

4K ROM

Enter the following lines:

```
10 FOR I=1 TO 300
20 PRINT "A";   (shift A)
30 NEXT I
```

Get a small AM radio and tune it for best response. Then press RUN and NEWLINE. Note the sound as you press NEWLINE. The display may be ignored.

Our thanks to:

Bill Phillips
Box 571
Ely, NV 89301

8K ROM

Enter the following lines:

```
10 FOR N=40 TO 60
20 FOR I=1 TO N
30 LET R=I*22/N
40 LET T=PI*100*I/N
50 PLOT 32+R*COS T,22+R*SIN T
60 NEXT I
70 PAUSE 100
80 CLS
90 NEXT N
```

Press RUN and ENTER. Observe the results.

Our thanks to:

Richard Booth
12875 Highland Rd.
Highland, MD 20777

Glitchoidz Report

Perceptions (1:6, p. 12, col. 3)
9110 LET TEMP=-1

Try This (2:1, p. 15)

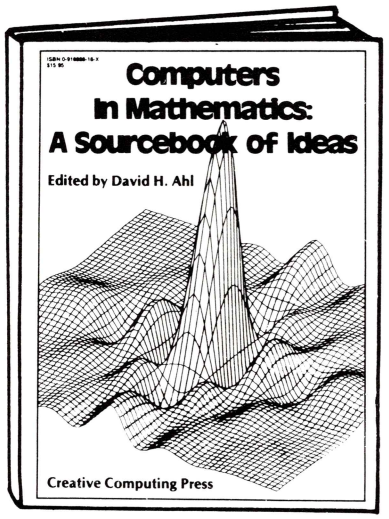
Both programs are best without the RAM pack.

Second program, line 10: put a) after the 2.

The Game of Life (2:1, p. 21, col. 3)
3) POKE 16435,20
240 IF A< 1 OR A > 300 . . .

Sourcebook of Ideas

Many mathematics ideas can be better illustrated with a computer than with a text book.



Consider Baseball cards. If there are 50 cards in a set, how many packs of bubble gum must be purchased to obtain a complete set of players? Many students will guess over 1 million packs yet on average it's only 329.

The formula to solve this problem is not easy. The computer simulation is. Yet you as a teacher probably don't have time to devise programs to illustrate concepts like this.

Between grades 1 and 12 there are 142 mathematical concepts in which the computer can play an important role. Things like arithmetic practice, X-Y coordinates, proving geometic theorems, probability, compounding and computation of pi by inscribed polygons.

Endorsed by NCTM

The National Council of Teachers of Mathematics has strongly endorsed the use of computers in the classroom. Unfortunately most textbooks have not yet responded to this endorsement and do not include programs or computer teaching techniques. You probably don't have the time to develop all these ideas either. What to do?

For the past six years, *Creative Computing* magazine has been running two or three articles per issue written by math teachers. These are classroom proven, tested ideas complete with flowcharts, programs and sample runs.

Teachers have been ordering back issues with those applications for years. However, many of these issues are now sold out or in very short supply.

So we took the most popular 134 articles and applications and reprinted them in a giant 224-page book called *Computers in Mathematics: A Sourcebook of Ideas.*

Ready-to-use-material

This book contains pragmatic, ready to use, classroom tested ideas on everything from simply binary counting to advanced techniques like multiple regression analysis and differential equations.

The book includes many activities that don't require a computer. And if you're considering expanding your computer facilities, you'll find a section on how to select a computer complete with an invaluable microcomputer comparison chart.

Another section presents over 250 problems, puzzles, and programming ideas, more than are found in most "problem collection" books.

Computers in Mathematics: A Sourcebook of Ideas is edited by David Ahl, one of the pioneers in computer education and the founder of *Creative Computing.*

The book is not cheap. It costs $15.95. However if you were to order just half of the back issues from which articles were drawn, they would cost you over $30.

Satisfaction Guaranteed

If you are teaching mathematics in any grade between 1 and 12, we're convinced you'll find this book of tremendous value. If, after receiving it and using it for 30 days you do not agree, you may return it for a full refund plus your return postage.

To order send payment* plus $1.00 postage and handling to Creative Computing, Dept. Z0422, 39 E. Hanover Avenue, Morris Plains, N.J. 07950. Orders may also be charged to your Visa, Mastercard or American Express account — by mail include credit card name, number and expiration date, or if you prefer, use our 24-hour toll-free number, 800-631-8112. In N.J. only 201-540-0445.

*N.J. Residents add 5% tax.

creative computing

4K/8K ROMs in One ZX 80

Michael Rubesch

Ed.—**A WORD OF CAUTION:** *Any hardware project that involves modifications to your computer must be approached with extreme caution.* **SYNC** *cannot be responsible for problems that may arise from attempting hardware projects. Obviously, any damage done to the computer can be costly in time and money for repairs or even replacement.*

I was very pleased with the new Basic that my ZX80 ran after I installed the new 8K ROM. The command set was much closer to "normal" Basic and all commands were available as single keystrokes. This was a considerable improvement over the 4K ROM.

However, as soon as I was finished testing the new features of this ROM, I was very disappointed to discover that my old 4K programs were not directly compatible with the new 8K system. In the first place, the tape standards were completely different, so that all programs would have to be entered from the keyboard. Secondly, not all commands were directly transferable so that in several cases I would have to edit and debug programs that already ran on the 4K system. Worst of all, the machine language programs that I had purchased were worthless since I could not simply enter them through the keyboard. I felt that I had to have the option of using either the 4K or the 8K ROM on the same ZX80.

My first idea was simple—install the appropriate ROM as needed: use the 8K for all new programming efforts, and the 4K for any old programs I might run. After taking the computer apart several times to get at the ROMs, and bending more than a few pins, I knew that idea was *too* simple.

This article describes the final solution that I used to make both ROMs available to my ZX80. I installed the ROMs piggyback in the socket and switch selected the ROM to be used. The modification is very simple and requires only a soldering iron, an spdt switch, and a few inches of wire.

The completed 4K/8K piggyback ROM assembly is shown in Figure 1. The first step to making the assembly was to remove the ROM from the ZX80. Pin 28 on each ROM was bent up slightly so that it was at a 45 degree angle to the other pins. (Pin 28, incidentally, is the pin to the right of the notch in the top of the ROMs.) This is the power pin of the ROMs; if power is not applied to the ROM, the rest of the system does not know it is in the circuit—even if all other pins are connected. By switching the power to only one of these pins only one ROM will be active at any time.

Except for pins 28, all of the other pins of the ROMs were connected in pairs by soldering them together. This was done by placing one ROM on top of the other (piggyback) with the notches at the same end. Each pin on the top ROM just overlapped the corresponding pin on the lower ROM by about one-tenth of an inch, and the only contact was between pairs of pins. It makes no difference which ROM is on the top, but I put the 8K one there.

Figure 1.

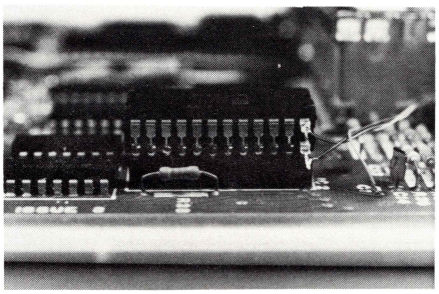
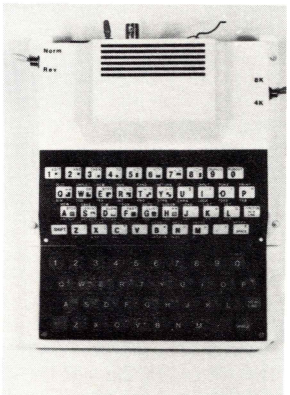


Figure 2.



The next step was the hardest one—the pairs of pins were soldered together. Each pair of pins was carefully heated with a soldering iron and a small amount of solder was applied where the pins overlapped. The solder was applied carefully so that no solder bridges were made between adjacent pairs of pins. Care was also taken to avoid getting solder on the tips of the pins of the lower ROM since they would be inserted in the socket of the ZX80. The bent pins (pins 28) were not soldered together.

A five inch piece of wire was soldered to the tip of each bent pin 28. The pin and wires were carefully oriented so that they did not touch each other. The wire from one of the bent pins was attached to each of the switched terminals of the spdt switch. A third five inch piece of wire was soldered to the common terminal of the switch. The wires were twisted together to keep them from tangling too much.

The piggyback ROM assembly was installed in the ROM socket of the ZX80. The bent pin pair was not inserted in the socket. The free end of the wire attached to the common terminal of the switch was soldered to +5V at the trace that leads away from pin 28 of the ROM socket. This trace can be seen to the right and above the ROM socket in Figure 1. The switch was mounted at a convenient location on the right edge of the case, and the case was reassembled. I found the greater height of the ROM assembly caused a slight gap in the ZX80 case, but I did not feel it was bad enough to warrant removing the socket and soldering the ROMs in place.

The combined 4K/8K ROM was easy to test. I applied power to the ZX80 and typed a zero. The 8K ROM displays a zero with a slash through it while the 4K ROM zero has no slash. I made sure that each switch position selected one of the ROMs and labelled the switch. The final step in the assembly was to put the 8K keyboard overlay above the original keyboard (as shown in Figure 2) so that I could refer to the appropriate key legends for each ROM.

I now have a much more flexible ZX80 than is available with just one ROM. None of the 4K programs in which I invested either time or money will be wasted, and yet I am able to use any new 8K program that I might obtain. I also never have to worry whether a program I find published in *SYNC* will be compatible because I have both ROMs available at a flip of a switch. ■

Michael Rubesch, Box 719 Chemistry, Purdue University, West Lafayette, IN 47907.

Getting Loaded

E. Ross Helton

When I tried LOADing my first program on my ZX80, I was excited. This excitement quickly changed to disappointment after my fifth or sixth attempt. Finally after several more tries, the program was LOADed. I marked the volume control. By the time I had loaded two or three more programs it was evident that even with a marked volume control, LOADing would and could still be difficult.

In searching for a better method, I came across an article in *SYNC* on using an LED for a loading monitor attached to the ZX80 internally.

If an LED could work inside, why not outside? Further study and experimentation proved that it could work! Externally mounted—the monitor was easy to alter, experiment with, and throw away if it proved unreliable. Also my ZX80 would not have to suffer unnecessary exposure of its vital parts if it did not have to be opened.

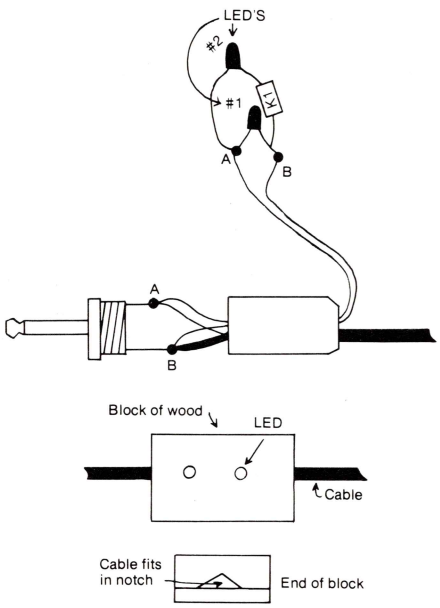
The monitor consists of two LEDs, a diode, some wire, and a case to house everything. The number 1 LED acts as an indicator of too low a volume or of the correct volume. The number 2 LED is an indicator of volume which is too loud.

To use the monitor, you should set your tape recorder to the portion just prior to the actual program. Plug the monitor into the earphone of the recorder, but do not plug into the ZX80 yet. Play the program. Watch the LEDs. The number 1 LED will glow brightly when the volume is correct; it will not light or it will barely flicker when the volume is too low. If the second LED is lit or flickering then the volume is too high for loading. When the monitor is adjusted so that only the number 1 LED is glowing brightly, rewind the tape, plug into the ZX80, and LOAD!

E. Ross Helton, 1408 N. 4th Ave. E., Newton, IO 50208.

Parts Needed (Radio Shack part numbers)

2 Subminiature LEDs # 276-1123
1 Micro miniature diode # 276-1104
1 piece of dual speaker wire
2 pieces of wood



To build the monitor, you need the items listed on the diagram (or equivalent). All of these items can be purchased at electronics supply houses. (As a point of reference, I have given the Radio Shack numbers.) Now decide which of the plugs on the loading cable you want to use for attaching your monitor. Unscrew the plastic cap (do not unhook any wires) and expose the two prongs inside. The long prong of the plug is the grounded side which I have labelled as "B" in the diagram. "B" is also the marking for the short lead of the LED (cathode side). Cut a piece of speaker wire about 3 to 6 inches long (the length is up to you). After stripping the two wires on one end, insert the wire through the back of the plastic cap of the plug in the same manner as the cable is inserted. Solder one wire to the short prong of the plug and the other wire to long prong (as illustrated). Now solder "A" wire to the long lead of the LED. Solder the other wire to the short lead (you may decide to shorten both the leads, so remember to keep track of the cathode which is to be connected to "B" wire). Solder the long lead of number 2 LED to the long lead of number 1 LED. Now solder the diode to the short lead of number 1 LED (make sure the cathode side is soldered to this lead—usually a color band indicates the cathode end). Connect other end of the diode to the remaining lead of the number 2 LED.

Screw the plastic cap back on the plug and test the monitor in the manner described. You should be able to load easily. Using a diode different from the one listed below will change the degree of brightness of both LEDs.

The case for the monitor can be constructed from thin crating wood or similiar material. Decide how big to make it (mine was 1" by 1-1/2"). Space and drill two holes for the LEDs. Hollow out the bottom side with a pocket knife or router. In the hollow space nest the LEDs and the diode (the tops of the LEDs should protrude above the top surface of the wood after being inserted from the bottom). Notch the bottom side of the block on either end (see diagram) deep enough to allow the cable to fit snug across the block. Cut another piece of wood to the same size. Screw it on the bottom side for a completely enclosed case. Judging the brightness of the LEDs may be easier if you paint the monitor case a dark color.

The monitor is now set: It will make little difference whether the monitor plug is hooked to the recorder or to the ZX80.

A further benefit of the monitor is in searching for a program on a tape. If the monitor is plugged into the recorder, you will see both LEDs flash during voice portions of the tape (a steady bright glow indicates a program). Another benefit is that it works with the ZX81 as well as with ZX80.

*L*oading is *E*asily *D*one with your new monitor!

Nine Defenders Against the Aliens

Martin Wren-Hilton

SYNC

SOFTWARE PROFILE

Name: Defender (Version 1.81)

Type: Arcade Space Fantasy

System: Sinclair ZX81, XZ80 (both 8K and 4K ROMs) (UK television, 625 lines only, at the moment)

Format: Cassette

Language: Z80 machine code

Summary: Better than any other arcade game I've seen.

Price: £5.50

Manufacturer:
Quicksilva
95 Upper Brownhill Rd.
Maybush, Southampton, Hants.
United Kingdom

I was absolutely amazed when I saw the *Defender* program for the first time. Written entirely in machine code, this program is better than any other arcade game I've seen.

After loading the game from the cassette, I glanced through the instruction sheet. This sheet tells the player which buttons to press ("6" to move down, "7" to move up, "9" to thrust forward and "0" to fire) and which addresses to POKE to alter the horizontal and vertical hold of the picture.

Entering RUN, the screen goes blank for three seconds, then the display appears: at the top of the screen is the number of Defender spaceships you have and your score, at the bottom of the screen three lines of 'moving scenery' give the effect of movement, and on the left hand side of the screen is your spaceship. It should be noted that this program only works on UK 625 line television at the moment, although the author is working on an American 525 line television version. This is because the screen display occupies the whole of the television screen from the very top right to the very bottom.

After a short period of time the aliens appear from the right. The aim of the game is to blast them to pieces without getting blown up yourself. You start with 9 Defender spaceships and lose one when you get hit. The aliens fire from right to left and can have up to six missiles on the screen at once. You get 100 points for each alien. As the game progresses, more and more aliens appear on the screen up to a hectic maximum of 8 aliens, each firing six missiles at you. Your Defender spaceship can have up to six missiles on the screen at once.

The general movement of the aliens is from right to left, and up or down depending upon the type of alien. Unlike the original arcade *Defender* by Williams, there are no Humanoids, Smart Bombs, Baiters, Bombers, Mutants, Pods or Attack Waves nor are there 'Reverse' or 'Hyperspace' buttons. There is no provision for high scores either.

Having said that, if you happen to have the Quicksilva Sound Board, this program generates some fantastic sound effects for phasers and missiles, and every time you hit an alien a brilliant sound effect is produced.

Defender is available for 4K ROM machines with UK 625 line television as well as this version for 8K ROM machines.

Defender is a difficult and highly entertaining game that completely fills the screen and produces brilliant sound effects. This program is highly recommended and will keep you and your friends entertained for many, many hours.

Machine Language Programming Made Simple

Martin Wren-Hilton

Machine Language Programming Made Simple For Your Sinclair ZX80 and ZX81, Melbourne House (Publishers) Ltd., Glebe Cottage, Glebe House, Station Road, Cheddington, Leighton Buzzard, Bedfordshire LU7, England. 160 pp. ringbound. £8.95.

Machine Language Programming Made Simple is an excellent introduction to the complicated and bewildering world of Z80 machine code. This book eases you through a course of lessons with diagrams, examples, and exercises. Users of both the ZX80 and ZX81 will find this book to be of great value because both machines are based upon the Z80A Central Processor Unit which is a faster version of the Z80 Central Processor Unit (or CPU for short).

The book is organized into four chapters and one appendix. The first chapter, "Finding your way around machine language," has seventeen sub-headings and discusses the way computers work, the concepts of binary and hexadecimal notation, counting up and down, logical operators (and how to use them), loops, jumps and block operators. Details are given on the uses of the twenty or so registers along with a discussion of what the stack is.

"Less frequently used instructions," the next chapter, explains Register exchanges, Bit, Set, Reset, Rotates and Shifts and In and Out in great detail. Numerous worked examples are provided.

Chapter three, "Programming your Sinclair," shows the user how powerful and fast low level languages are, how to plan a machine language program, and, once written, how to debug and edit it. Four different methods of program storage with their comparative advantages are given.

The final chapter contains four programs related to machine language programming—*Displaying 100 Bytes in Hex, Machine Code Editor, Loading Code into Array* and *ZX81 Draughts*. The first three programs are referred to throughout the book; the fourth is the same as *Draughts* in *Not Only 30 Programs*, also by Melbourne House.

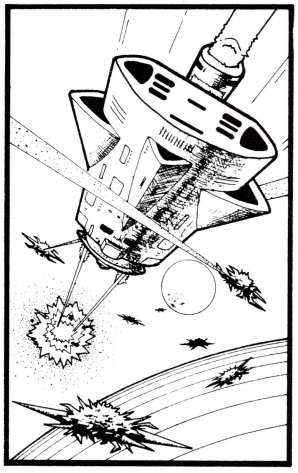
Machine Language Programming Made Simple is a clear, concise volume which makes an excellent introduction to Z80 machine code. The price is a little heavy at £8.95, but this might be expected of a specialist book like this. To summarize, this book is highly recommended to the beginner. ■

QUICKSILVA

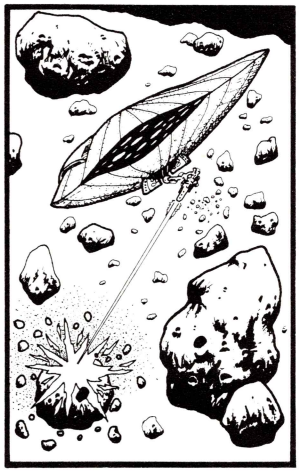
Presents a range of top quality hardware & software for the

ZX-81 and ZX-80

The very best in machine code fast moving graphics arcade games. Cassette Inserts are full colour prints of original art work by 'STEINARLUND'.

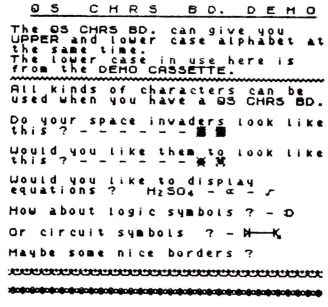


QS DEFENDER.
UP - DOWN - THRUST - FIRE
First and only full screen display. Software to drive QS SOUND BD. Moving Planetary surface. Up to 84 fast moving characters on screen at once. On screen scoring. Ten missiles at once. Increasing attack patterns. Requires 8K ROM and 4K min of RAM.



QS ASTEROIDS
Quicksilva's new arcade game. LEFT - RIGHT - THRUST - FIRE Software to drive QS SOUND BD. Multiple missiles firing in 8 directions. On screen scoring. Increasing number of asteroids. Full mobility of ship to all areas of the screen. Two asteroid sizes. Bonus ship at 10,000 points. Requires 8K ROM, 4K min of RAM + SLOW function.

QS CHRS BD./
A programmable character generator giving—128 SEPARATELY PROGRAMMABLE CHARACTERS. ON/OFF SWITCH. 1K ON BOARD RAM. Enables creation and display of your own characters to screen or printer. Demo cassette of fast machine code opeation routines and lower case alphabet included. See below for ZX PRINTER listing.


QS CHRS BD. DEMO
The QS CHRS BD. can give you UPPER and lower case alphabet at the same time.
The lower case in use here is from the DEMO CASSETTE.
All kinds of characters can be used when you have a QS CHRS BD.
Do your space invaders look like this ?
Would you like them to look like this ?
Would you like to display equations ? H_2SO_4
How about logic symbols ?
Or circuit symbols ?
Maybe some nice borders ?

QS HI-RES BD.
A Hi-res graphics board giving—256 x 192 PIXELS. 6K ON BD. RAM. SOFTWARE SELECT/DESELECT. MIXED TEXT AND GRAPHICS. 2K ON BOARD ROM. Resident fast machine code graphics software (in ROM) provides the following HI-RES Commands.—MOVE x,y; PLOT x,y; DRAW x,y; PRINT X$; COPY; BLACK; WHITE; CLEAR. See side for ZX PRINTER listings using COPY.

QS SOUND BD.
A programmable sound effects board using the AY-3-8910. 3 TONES; 1 NOISE; ENVELOPE SHAPER: + TWO 8 BIT I/O PORTS. Easily programmable from BASIC, the AY chip does most of the work leaving your computer free for other things. Signal O/P via 3.5 mm Jack socket Ports O/P via a 16 pin I.C. Socket.

QS MOTHER BD. & QS CONNECTOR.
A reliable expansion system allowing a total of any RAM pack plus two other plug in boards to be in use at once. On board 5 V regulator drives all external boards. Fitted with two 23 way double sided edge connectors. Connector is 2 x 23 way edge conns soldered back to back. Expansion can operate in two ways
1) COMPUTER↔CONNECTOR↔Any QS add on bd. (but no extra ram pack)
2) COMPUTER↔CONNECTOR↔MOTHER BD↔ANY RAM PACK. (2 bds to fit in mother Bd.)

QS RAM BDS.
Two sizes of RAM Bds are available. A 3K static RAM bd (no case) and a fully cased 16K dynamic RAM BD. Both are extremely reliable and will fit any ZX COMPUTER.

ALL PRODUCTS FULLY GUARANTEED. FULLY INCLUSIVE PRICES ARE AS FOLLOWS—
QS DEFENDER £6.00; QS ASTEROIDS £6.00; QS LIFE £5.00; QS CHRS DEMO £4.00; QS MOTHER BD. £13.00; QS CONNECTOR £5.00; QS CHRS BD. £27.00; QS SOUND BD. £27.00; QS HI-RES BD. £87.00; QS 3K RAM £20.00; QS 16K RAM £35.00.
All payments in sterling (ie. International Moeny Order, Bankers Draught). Discounts as follows—over £30 subtract £1; over £60 subtract £2.50; over £90 subtract £5. Catalog 50 pence. Orders and enquiries to QUICKSILVA : 95, UPPER BROWNHILL RD. : MAYBUSH : SOTON : HANTS : ENGLAND. Please state Type of machine, Which ROM, Memory size, when ordering.

The Exploding Bookshelf

Eric Deeson

Considering that a couple of years ago no one had heard of Sinclair/MicroAce computers, the number of books published about them in the UK is quite astonishing.

As the organizer of Britain's second largest ZX user group, I have so far accumulated thirty of these publications (sent for review mainly). So it is certainly time for some kind of overview.

The first level of overview is that of the kite, as complete a list as possible of books published (or forthcoming) on the ZX80 and/or ZX81. (This list, by the way, is drawn from the first *Directory of Suppliers to the ZX Market*, recently published by my group. The directory gives details of some 250 product lines from 150 suppliers (mainly British).

British Publishers of ZX-Specific Books

Artic Computing, 396 James Reckitt Avenue, Hull.

The ZX80 ROM (assembly listing of the Monitor).

Bug-byte, 98-100 The Albany, Old Hall Street, Liverpool 3.

The ZX80 Programming Course (with cassette).

Computer Publications, 3/33 Woodthorpe Road, Ashford, Middlesex.

Making the Most of Your ZX80 (mainly annotated games listings).

Stretching Your ZX80 or ZX81 to Its Limits (ditto).

Cooksey, 31 Haslingfield Road, Harlton, Cambridge.

Over 100 Programs for Beginners on the ZX81 (how to write programs for elementary school math).

Database Consultancy, 105 Fairholme Avenue, Gidea Park, Romford, Essex.

Interface—the Early Days (extracts from the first ZX periodical).

The Gateway Guide to the ZX81 and ZX80 (mainly annotated games listings). (in U.S. from Creative Computing.)

Getting Acquainted with Your ZX81 (ditto). (In U.S. from Creative Computing.)

Mastering Machine Code on Your ZX81 or ZX80 (see below).

Educare, 139a Sloane Street, London SW1.

Fifty 1K Programmes for Primary Education (simple stuff for the early grades).

Richard Francis, 22 Foxhollow, Barhill, Cambridge.

The Cambridge Collection (30 annotated 1K ZX81 listings; cassette available).

Fylesoft, 114 Harris Street, St Helens, Merseyside.

Fortune Teller (a number of 1K ZX81 listings for the occult).

Griffin + George, Ealing Road, Alperton, Middlesex.

About Computers (see below).

Hewson Consultants, 7 Grahame Close, Blewbury, Oxford.

Hints and Tips for the ZX80 (much good material here).

Hints and Tips for the ZX81 (see below).

ICL, 60 Portman Road, Reading, Berks.

How to Program the ZX81 (long announced but not yet available).

Interface, 44/6 Earls Court Road, London W8.

50 Rip-roaring Games for the ZX80 and ZX81 (listings).

30 Amazing Games for the 1K ZX81 (ditto).

Linsac, 68 Barker Road, Middlesbrough, Cleveland.

The ZX80 Companion (cassette available).

The ZX81 Companion (ditto) (see below). (In U.S. from Creative Computing.)

Longman (offices in US).

The ZX81 in Teaching (survey of uses with lots of tips).

Macmillan (offices in US).

The Sinclair ZX81 (cassette available) (see below).

Melbourne House, Glebe House, Station Road, Cheddington, Beds.

30 Programs for the ZX80 1K (games).

30 Programs for the ZX81 1K (games).

Machine Language Programming for Your Sinclair (see below). (In U.S. from Softsync.)

Understanding Your ZX81 ROM (see below). (In U.S. from Softsync.)

Complete ZX81 BASIC Programming Course (cassette available).

ZX81 Monitor Listing (inside the ROM).

MJC Publishing, c/o BCM Primal, London WC1C 6XX.

The Giant ZX81 Programming Book (collection, not yet available).

Newnes (offices in US).

Learning BASIC with Your Sinclair ZX80 (very good).

Learning BASIC with Your Sinclair ZX81 (not yet available).

Phipps Associates, 3 Downs Avenue, Epsom, Surrey.

The ZX80 Pocket-book (very good basic guide).

The ZX81 Pocket-book (see below). (In U.S. from Gladstone.)

Shiva Publishing, 4 Church Lane, Nantwich, Cheshire.

PEEK, POKE, Byte and RAM! (see below).

Using the 16K ZX81 (not yet available).

Sigma Technical Press, c/o John Wiley (offices in US).

Byting Deeper into Your ZX81 (a fairly thorough guide).

Sinclair Research (offices in US).

ZX81 Learning Lab (not yet available).

Spencer, The Sycamores, Queen's Road, Hodthorpe, Nottingham.

The Hodthorpe Collection (educational ZX81 skeletons; cassette available).

Timedata, 57 Swallowdale, Basildon, Essex.

The ZX80 Magic Book (with ZX81 supplement; a nice early work).

Usherwood, 53 Marlborough Road, Stockton, Cleveland.

20 Programs for Your ZX80 (mainly 1K games).

Eric Deeson, Highgate School, Birmingham B12 9DS, UK.

Wood, 22 Great Coates Road, Grimsby, Humberside.

Beginning Micro-computing with the ZX81 (brief, cheap BASIC guide).

Zipprint, 418 Poole Road, Parkstone, Poole, Dorset.

ZX80 Programs Volume 1 (there wasn't a Volume 2—).

Quite a number of these publishers have offices, agents, or representatives in the United States. That is why I give no indication of prices in this survey. Scan the pages of *SYNC* for such information, but, if you fail, you will get speedy replies from the addresses in Britain.

Assessments

Now the kite swoops down to a much lower altitude. I would like to give paragraph assessments of what seem to me to be the most useful books for the 8K ROM machine. Some have been reviewed in detail in these pages already—but I think it is worth discussing them in the general context offered by this feature. The order in which the books are discussed is *not* random—I have tried to put them along a spectrum of difficulty.

PEEK, POKE, Byte and RAM! by Ian Stewart and Robin Jones (Shiva Publishing) is a quite delightful book for beginners. Produced by one of Britain's smaller professional publishers, this book has more visual impact than the rest put together. It is intended to replace the ZX81 manual to some extent—after all, the manual may be better than most but it is still far from perfect. These authors put themselves at the disposal of the real beginner—and beguile him/her with a wealth of jokes and cartoons as well as a light but definitive text. Definitely a best buy for those still struggling with the Manual.

If you have passed that stage, you need something to stretch you a bit more, but without pain. Phipps' *ZX81 Pocket-book*, by Trevor Toms, is just the job. It is an efficient guide to the more sophisticated aspects of ZX81 Basic; using the string functions and efficient programming are two that appear fairly early on. The book is spiral bound and well-laid out.

Andrew Hewson's *Hints and Tips for the ZX81* (Hewson Consultants) is much more obviously a home-brew publication. It has no visual impact and is fairly hard to wade through. All the same, if you have the patience, you will find many valuable "hints and tips" in these pages. Patience *is* required—the author's style and the publisher's layout are not conducive to easy reading or reference.

Bob Maunder is well known to *SYNC* readers; so probably too is his *ZX81 Companion* (published by Linsac and Creative Computing). This is the first book on this list to concentrate seriously on ZX81 programming applications. While Maunder naturally does not (cannot) avoid some discussion of Sinclair Basic, such discussions always appear in the context of applications. The first chapter explores the use of PRINT and PLOT in static and dynamic displays. Information processing and educational programming are the other major areas considered in thoughtful detail and depth. The final brief chapter is on the 8K Monitor (and gives a listing).

"Programming for real applications" is the subtitle of Randle Hurley's book *The Sinclair ZX81* (Macmillan). Clearly Hurley has the same idea of ZX81 development as Maunder. In practice, however, the books are very different. Hurley's comes from a "real" publisher, but is, in fact, a lot inferior as regards layout and appeal (no illustrations for instance). His approach, all the same, is excellent. After a good introductory dissertation on memory/time saving (with lots of quantitative data), he gives us a number of "serious" 16K programs with full discussion. (A cassette is available to save all that keying.) The programs include a sort of text editor (with a readability test routine for teachers preparing worksheets) and a technique for storing 18,000 items of data in the 11½K left by the program. Clever! There are financial and educational administration programs, too. Pity about the poor presentation, but, of course, "real" publishers are not used to working at high speed.

As the author of *About Computers* (Griffin + George), I must now be objective rather than give opinions. This book is explicitly for teachers and attempts to meet three objectives. First, it intends to be a gentle course on computer awareness, with "real life" applications illustrated by ZX81 routines. Second, the book provides a graded introduction to ZX81 Basic. And third, there are lots of ideas for computing projects. Various appendices conclude the book. This book then is explicitly about using the micro in a professional context.

Most ZX81 authors and writers tend to steer clear of machine coding—many even claim, quite wrongly, that it is difficult and pointless. However, of the quarter million ZX81 owners, a good number have long demanded definitive guides to this major subject. To help meet that demand, three good books have recently appeared.

The most detailed is Toni Baker's *Mastering Machine Code on Your ZX81 or ZX80* (the "or ZX80" is in fact in very small print in the title). Published by Database Consultancy, this is not brilliantly produced. However, it *is* brilliantly written. In nearly 200 close-packed pages the author guides one effortlessly along the track to full m/c code confidence. She does so with care and with firmness, providing all the time a good pressure towards progress. This is not an easy book because the subject is not easy. Lock yourself away for a week with Toni Baker's book and the ZX81 as chaperone, and you will be amazed at what you can do.

Machine Language Programming Made Simple and Ian Logan's *Understanding Your ZX81 ROM* are far less detailed introductions. Both are published by Melbourne House—well written but the printing and proofreading need considerable attention. The former book provides the elements of machine coding with a very gentle introduction on basic principles. The second, subtitled "Using machine code in your BASIC programs," has the neat approach of using examples from the 8K Monitor to illustrate the techniques used. However, neither book really encourages/forces you to try out the new ideas met. Toni Baker's does, and that is a telling advantage.

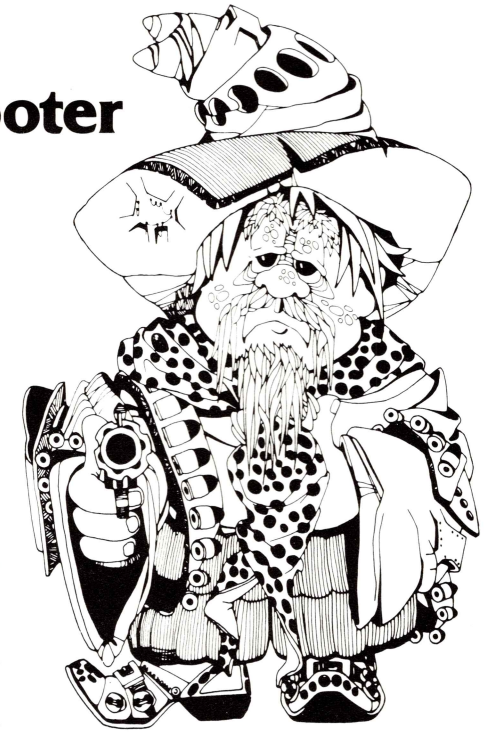
The ZX81 is past its honeymoon period in Britain now, and is coming to be recognized as a powerful general-purpose machine in its own right. The use of 48K RAM with it is becoming common (and packs giving up to 132K are available). We have the printer just about debugged, colour and sound add-ons are advertised, and disc systems should be with us by the time you read this.

In fact, there is an explosive growth in hardware add-ons as well as in books. That is the next big challenge for the ZX81 and one book explicitly about this field is due to appear shortly.

To me, the ZX81 was a milestone in the development of personal computing. The exponential sales and the proliferation of publications confirms that view. ■

8K ROM
16K RAM

6 Shooter



Larry G. Dighera

6 Shooter is a game of skill in which you attempt to hit a randomly moving target. When the target appears in a position which you feel is in your sights, you pull the trigger (hit a key). This causes a bullet to be fired, and, if the target is hit, it disintegrates in accordance with the accuracy of the shot. The number of shots fired and hits scored are displayed at the bottom of the screen. After six shots you must reload to continue playing. Before the commencement of play, you are asked to select the level of difficulty you desire. As your skill increases, you may increase the level of difficulty and continue to find the game challenging. Indeed, at skill factor 0, success requires lightning fast reflexes. Because play occurs in real time, through the use of the INKEY$ function, I find a level of excitement often absent in other ZX81 games.

Many of the techniques used in *6 Shooter* may be adapted for use in your programs. *6 Shooter* has been optimized for speed by placing often used subroutines at the beginning; the operating system searches for called subroutines beginning at the first program line. The subroutine which produces the title in characters four times their normal size is easily alterable to display any message by changing the contents of T$.

The algorithm which moves the bullet across the screen is conveniently short, and the apparent speed may be controlled by changing step value.

Another useful subroutine is found at RICOCHET. Virtually any pattern of movement may be created by the use of this routine.

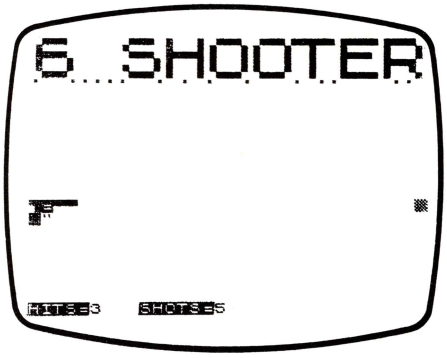
A Basic DATA statement is created by allowing R$ to hold the coordinates to be PLOTed and READing them out with the FOR, NEXT loop. By the use of PRINT AT statements, selected areas of the screen may be written over for updating or erased by over printing with spaces, as in the CLS and score subroutines.

Larry G. Dighera, P.O. Box 12100, Santa Ana, CA 92712.

With 16K of user programmable memory available, there is no reason why a creative individual could not expand this program to include such features as: automatically increasing the difficulty with each reload, score keeping for two players, different types of targets with different point values, linear vertical movement of the target. Another nice feature would be to ask the player who has beaten the previous high score to enter his name as the new champ.

Good luck and good shootin'. ■

Sample Run



Listing

```
 10 REM **6 SHOOTER**
 20 REM COPYRIGHT 1982  REV.1.0
 30 REM DIGHERA
 40 FAST
 44 GOTO 1300
 50 REM CLS
 51 GOSUB 1150
 52 PRINT AT TRG-1,30;" "
 53 PRINT AT TRG+1,30;" "
 55 PRINT AT TRG,31;" "
 57 PRINT AT 13,31;" "
 59 RETURN
 60 REM TARGET GENERATER
 80 PRINT AT TRG,31;" "
 90 LET MISS=0
100 LET TRG=INT (RND*17)+4
110 PRINT AT TRG,31;"▒"
120 RETURN
130 REM SKILL
140 IF INKEY$<>"" THEN RETURN
150 FOR W=1 TO SKL
160 IF INKEY$<>"" THEN RETURN
170 NEXT W
180 LET MISS=1
190 RETURN
200 REM SCREEN
220 LET X1=0
230 LET T$="6 SHOOTER"
240 PRINT AT 3,0;"
.....
250 FOR I=1 TO LEN T$
260 LET J=CODE T$
270 LET T$=T$(2 TO LEN T$)
280 FOR Y=0 TO 7
290 LET K=PEEK (7680+J*8+Y)
300 LET L=128
310 FOR X=0 TO 7
320 IF K<L THEN GOTO 350
330 PLOT X+X1,43-Y
340 LET K=K-L
350 LET L=L/2
360 NEXT X
370 NEXT Y
380 LET X1=X1+7
390 NEXT I
400 PRINT AT 21,0;"HITS=0";"0
";
410 PRINT "SHOTS=0"
420 PRINT AT 13,0;"▀▀▀"
430 PRINT AT 14,0;"▀▀▀"
440 RETURN
450 REM SKILL SELECT
460 GOSUB 1130
470 PRINT AT 9,5;"SELECT SKILL
FACTER"
480 PRINT AT 10,5;"(0=DIFFICULT
, 10=EASY)"
490 INPUT SKL
500 PRINT AT 8,0
510 FOR I=1 TO 64
520 PRINT " ";
530 NEXT I
550 RETURN
850 REM ****TURN****
860 IF SHTS>0 THEN GOSUB 50
870 LET TRG=21
880 PRINT AT 13,0;"▀"
890 GOSUB 70
900 IF INKEY$<>"" THEN GOTO 960
910 IF TRG<>21 AND MISS<>1 THEN
 GOSUB 130
920 IF MISS=1 THEN GOSUB 70
940 IF INKEY$="" THEN GOTO 900
960 REM BULLET
970 FOR X=0 TO 63 STEP 9
980 IF X>0 THEN UNPLOT X-9,17
990 PLOT X,17
1000 NEXT X
1010 LET SHTS=SHTS+1
1030 IF TRG<12 OR TRG>14 THEN RE
TURN
1050 REM EXPLO
1060 PRINT AT TRG,31;"▒"
1070 PRINT AT TRG-1,30;"▒"
1080 PRINT AT TRG+1,30;"▒"
1084 IF TRG=13 THEN GOTO 1100
1085 REM RICOCHET
1086 LET R$="5819572155225322542
1552354255227502848284927502949 3
1"
1087 REM LET R$="601959215722552
2862157235625542752285028512752 2
95131"
1088 FOR I=1 TO 13*4 STEP 4
1090 IF I>1 THEN UNPLOT VAL R$(I
-4 TO I+1-4),VAL R$(I+2-4 TO I+3
-4)
1091 PLOT VAL R$(I TO I+1),VAL R
$(I+2 TO I+3)
1092 NEXT I
1096 UNPLOT VAL R$(I-4 TO I+1-4)
,VAL R$(I+2-4 TO I+3-4)
1100 LET HITS=HITS+1
1110 RETURN
1130 REM SCORE
1150 PRINT AT 21,5;HITS
1160 PRINT AT 21,15;SHTS
1180 RETURN
1300 REM ****UTILITY****
1330 GOSUB 200
1340 LET SHTS=0
1350 LET HITS=0
1360 GOSUB 460
1370 SLOW
1380 GOSUB 850
1400 GOSUB 1130
1420 IF SHTS<6 THEN GOTO 1380
1430 GOSUB 50
1440 REM RELOAD
1450 PRINT AT 9,5;"HIT L TO RELO
AD"
1460 PRINT AT 9,5;"HIT"
1470 PRINT AT 9,5;"HIT"
1480 IF INKEY$="" THEN GOTO 1460
1490 IF INKEY$="L" THEN GOTO 134
0
1500 IF HITS=0 THEN PRINT AT 9,5
;"YOU NEED PRACTICE"
1590 IF SHTS/HITS<=2 THEN PRINT
AT 9,5;"** GOOD SHOOTIN **"
1600 IF SHTS/HITS<=2 THEN PRINT
AT 9,5;"** GOOD SHOOTIN **"
1610 GOTO 1590
1900 STOP
2000 SAVE "6 SHOOTER"
2010 PRINT "LOAD COMPLETE"
2020 PAUSE 200
2024 CLS
2030 RUN
```

Isolation

Drew Nisbet

In *Isolation* you attempt to encircle one of the computer's men, isolating him from all of his cohorts. A man is isolated when you have placed one of your men above, below, to the left, to the right and in the four locations diagonally adjacent to the victim. Therefore, a man located in a corner of the playing area requires only three men to be cut off; a man on any other edge position will require five men to be isolated; and for any man located away from the edges you will need eight of your men to surround him. The computer may or may not place a new man on the board on its turn. By adding another man it may thwart your plans, or the man may fall right into your clutches! At the end of the game you receive a point for each of the computer's men you have isolated from his confreres.

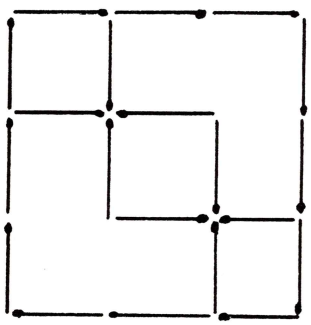
Drew Nisbet, 6 Moffatt Crt., Rexdale, Ont. M9V 4E1.

Press RUN and NEWLINE to start the game. One of the Computer's men (a gray square) will appear on the board. Select a position for your man (a black square) and enter it as follows: letter, number, NEWLINE. When all of the positions on the board have been filled, or, if you enter "0" (zero) to end the game, your score will be displayed.

```
 10 DIM A(81)
 20 LET T = 0
 30 LET S = 0
 40 FOR I = 1 TO 81
 50 IF I < 11 OR I > 71 OR (I / 9) * 9 = I OR
    ((I - 1) / 9) * 9 = I - 1 THEN LET A(I) = 1
 60 NEXT I
 70 FOR I = 1 TO 7
 80 PRINT CHR$(I + 37);"#";
 90 FOR J = 1 TO 7
100 PRINT "+";
110 NEXT J
120 PRINT
130 NEXT I
140 PRINT
150 PRINT "##1234567"
160 LET D = RND(61) + 10
170 IF A(D) > 0 THEN GO TO 160
180 LET A(D) = 2
190 LET Y = D / 9
200 LET X = D - 9 * Y - 1
210 LET M = 9
220 GO SUB 350
230 INPUT Y$
240 LET Y = CODE(Y$) - 37
250 IF Y = -9 THEN GO TO 400
260 LET X = CODE(TL$(Y$)) - 28
270 IF Y > 7 OR X > 7 THEN GO TO 230
280 LET C = 9 * Y + X + 1
290 IF A(C) > 0 THEN GO TO 230
300 LET A(C) = 1
310 LET M = 128
320 GO SUB 350
330 IF RND(3) = 2 THEN GO TO 230
340 GO TO 160
350 LET P = X + 2 + 10 * (Y - 1)
360 POKE PEEK(16396) + 256 * PEEK(16397) + P,M
370 LET T = T + 1
380 IF T = 49 THEN GO TO 400
390 RETURN
400 FOR I = 11 TO 71
410 IF (A(I) = 2 AND A(I - 10) = 1 AND A(I - 9) = 1
    AND A(I - 8) = 1 AND A(I - 1) = 1 AND A(I + 1) = 1
    AND A(I + 8) = 1 AND A(I + 9) = 1 AND A(I + 10) = 1)
    THEN LET S = S + 1
420 NEXT I
430 PRINT "POINTS =#";S
440 STOP
```

puzzle answers

A Square Problem:



Another Square Problem:

11	14	18	4
17	5	10	15
6	20	12	9
13	8	7	19

A Folding Puzzle: 11.57 inches is the length of the folded line AB.

A Square Deal: The cards are: (1) The four of hearts. (2) The six of hearts. (3) The six of spades.

A Snail's Pace: The answer is 13 days. After 12 days he was 12 feet up the wall of the well. On the 13th day he made it over the top and didn't have to slide back down again.

resources

Educational Software

- Elementary school level package, grades 1-4. Catalog program, 9 learning and 3 game programs. 8K ROM; 2K RAM. Full documentation, $14.95; cassette, $12.95; package of both, $24.95. Checks payable to:
  Edson Electronics
  PO Box 151211
  Tampa, FL 33684
  Phone: (813) 870-0282

Loading Hardware

- Loading is easy—with an LED monitor. Kit, $6 + $1 s&h.
  R. H. Enterprises
  1408 N. 4th Ave. E.
  Newton, IO 50208

- Digital display cassette loading monitor; hooks up in seconds outside the computer; helps insure proper loading level without guessing. $28.95 from:
  Edson Electronics
  PO Box 151211
  Tampa, FL 33684

Floppy Disk

- Disk file storage unit to interface to the ZX81. Shugart SA400 (or equivalent) 5 1/4 in. diskette drive; motherboard; power supply; cabinet; TTL interface; 10 sector diskettes. For further information contact:
  Macronics
  26 Spiers Close
  Knowle, Solihull
  West Midlands B93 9ES
  United Kingdom

Memory Expansion

- The Incremental 1.1 memory expansion system. Fits inside ZX81; plugs in, no soldering; up to 16K in increments of 2K by additional plugins; low power. Incremental 1.1 socketed for maximum of 8K, £8.50; for maximum of 16K, £9.50. SAE for details to:
  East London Robotics
  Finlandia House
  14 Darwell Close
  East Ham
  London E6 4BT
  United Kingdom

- Memory modification kit to increase the ZX81 to 2K RAM; $20 from:
  Cosmonics
  Box 10358
  San Jose, CA 95157

Interfacing

- RD 8100 System. Highly flexible modular interface system to permit real-time application of the ZX80/81. Connected by motherboard; 5 modules available starting at £27.50. For full details write:
  R. E. Dickens
  RD Laboratories
  5 Kennedy Road
  Dane End
  Ware, Herts, SG12 OLU
  United Kingdom

Utility Programs

- *Cassette I/O for ZX81.* Set of utility routines to selectively read or write strings or arrays to a cassette; 500 bytes; 2K RAM required; $20 from:
  Cosmonics
  Box 10358
  San Jose, CA 95157

Users Groups

- The West Los Angeles ZX80/81 Users' Group
  PO Box 34545
  Los Angeles, CA 90034
  Attention: Dr. George Kuby
  Phone: (213) 550-5035 (afternoons only)

- USA and International users welcome. Send self-addressed stamped envelope; indicate areas of interest, background with the ZX80/81, special needs, comments.
  ZX Users Group of New York
  PO Box 560 Wall Street
  New York, NY 10005

- Any one interested in forming a North Jersey Shore Users Group? If so, contact:
  Bill Thompson
  PO Box 427
  Rumson, NJ 07760

Fairs and Workshops

- Third ZX MICROFAIR. April 30 and May 1, 1982. Central Hall, Westminster, London SW1. For information exhibitors should contact:
  Mike Johnston
  71 Park Lane
  Tottenham
  London N17 OHG
  United Kingdom
  Phone: 01-801-9172

- "Microcomputers in Education" workshop series offered on April 16, 17, 18 at Jersey City State College in Jersey City, NJ; May 18, 19, and 20 at Gutman Library, Cambridge, MA; June 7, 8, 9 at Taft School in Watertown, CN. One day workshops; ten topics to choose from; hands-on experience. For further information write:
  Technical Eduction Research Centers
  8 Eliot St.
  Cambridge, MA 02138
  Phone: (617) 547-3890

Programs

- 8K/1K or 8K/2K games: *Basketball, Alien Destroyer, Beat the Maze*, and others. Listings, $1 + $.50 s&h. Cassette tape-one program, $6 + $1 s&h.
  R. H. Enterprises
  1408 N. 4th Ave. E.
  Newton, IO 50208

- Wide range of programs on cassette for ZX81, including: *Colditz*, £5.95; *Trafalgar, Microbox, Fighter*, and *Breakout*. £4.95 each. Last two avail in joystick version.
  Microx
  52 The Strand
  Worthing, Sussex
  United Kingdom

- *Astrology for the ZX81, 16K RAM (ZX80, 8K ROM 16K RAM).* Program casts accurate horoscopes in 3 min.; houses, planets, risings, and moon signs. Dealers/writers inquiries invited. Cassette and instructions, $20 (£12.50; U.S. bankdraft) from:
  Active Designs
  1108 Biltmore Dr.
  Nashville, TN 37204 ■

Make the Most of Your ZX81 or 80

The ZX81 Companion

The ZX81 Companion by Bob Maunder follows the same format as the popular **ZX80 Companion**. The book assists ZX81 users in four application areas: graphics, information retrieval, education and games. The book includes scores of fully documented listings of short routines as well as complete programs. For the serious user, the book also includes a disassembled listing of the ZX81 ROM Monitor.

MUSE reviewed the book and said, "Bob Maunder's **ZX80 Companion** was rightly recognized to be one of the best books published on progressive use of Sinclair's first micro. This is likely to gain a similar reputation. In its 130 pages, his attempt to show meaningful uses of the machine is brilliantly successful."

"The book has four sections with the author exploring in turn interactive graphics (gaming), information retrieval, educational computing, and the ZX81 monitor. In each case the exploration is thoughtfully written, detailed, and illustrated with meaningful programs. The educational section is the same—Bob Maunder is a teacher—and here we find sensible ideas tips, warnings and programs too."

Softbound, 5 1/2 x 8", 132 pages, $8.95.

Getting Acquainted With Your ZX81

This book is aimed at helping the newcomer make most effective use of his ZX81. As you work your way through it, your program library will grow (more than 70 programs) along with your understanding of Basic.

The book is chock full of games such as *Checkers* which draws the entire board on the screen. Other games include Alien Imploders, Blastermind, Moon Lander, Breakout, Digital Clock, Roller-Ball, Derby Day, and Star Burst.

But the book is not all games. It describes the use of PLOT and UNPLOT, SCROLL, arrays, TAB, PRINT AT, INKEYS, random numbers and PEEK and POKE. You'll find programs to print cascading sine waves, tables and graphs; to solve quadratic equations; to sort data; to compute interest and much more.

Softbound, 5 1/2 x 8", 120 pages $8.95.

The Gateway Guide to the ZX81 and ZX80

The Gateway Guide to the ZX81 and ZX80 by Mark Charlton contains more than 70 fully documented and explained programs for the ZX81 (or 8K ZX80). The book is a "doing book," rather than a reading one and the author encourages the reader to try things out as he goes. The book starts at a low level and assumes the ZX80 or ZX81 is the reader's first computer. However by the end, the reader will have become quite proficient.

The majority of programs in the books were written deliberately to make them easily convertible from machine to machine (ZX81, 4K ZX80 or 1K ZX80) so no matter which you have, you'll find many programs which you can run right away.

The book describes each function and statement in turn, illustrates it in a demonstration routine or program and then combines it with previously discussed material.

Softbound, 5 1/2 x 8", 172 pages, $8.95.

Computers For Kids, Sinclair Edition

Computers For Kids. by Sally Larsen is the fourth book in this highly successful series. (Previous editions have been released for TRS-80, Apple and Atari computers.) Written expressly for youngsters ages 8 to 13, the book requires no previous knowledge of algebra, variables or computers. Armed with a ZX81 and this book, a child will be able to write programs in less than an hour. A section is included for parents and teachers.

The book starts with a patient explanation of how to use the Sinclair, graduates to flow charts, and simple print programs. The twelve easy-to-read chapters go through loops, graphics and show other programming concepts, and show in a painless way how to make the computer do what you want.

Donald T. Piele, Professor of Mathematics at the University of Wisconsin-Parkside says, "**Computers For Kids** is the best material available for introducing students to their new computer. It is a perfect tool for teachers who are learning about computers and programming **with** their students. Highly recommended."

Softbound, 8 1/2 x 11", 56 pages, $3.95.

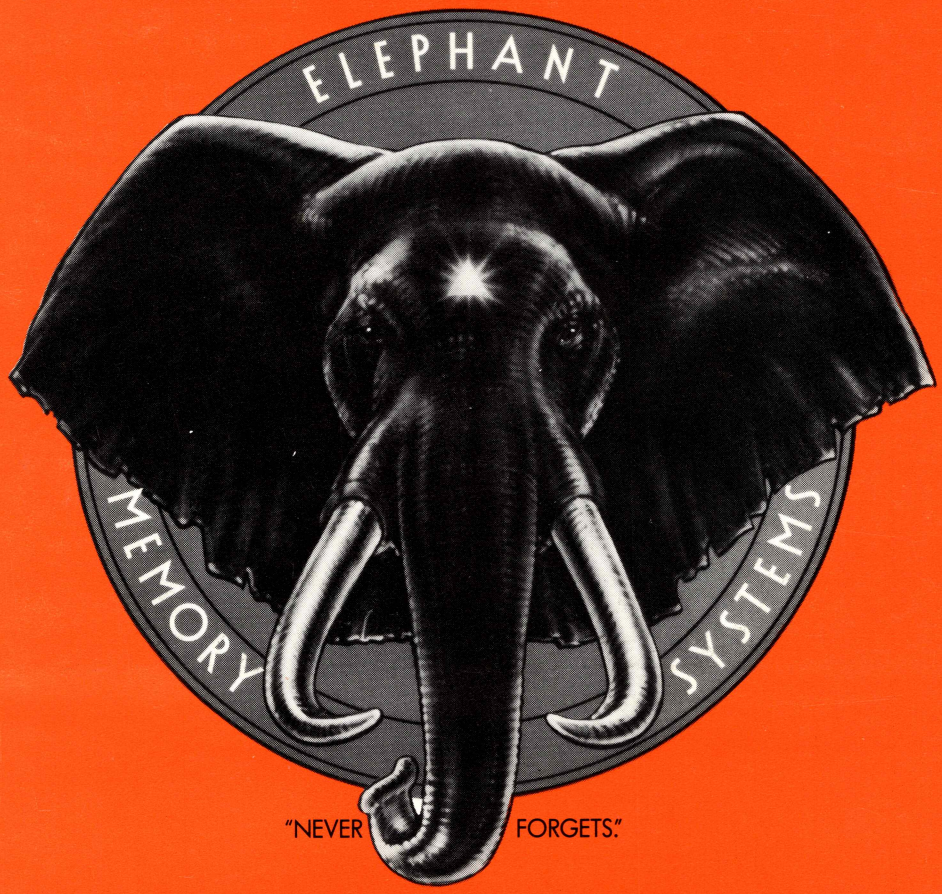
Order Today

To order send payment* plus $2.00 postage and handling to Creative Computing, Dept. Z0420, 39 E. Hanover Avenue, Morris Plains, N.J. 07950. Orders may also be charged to your Visa, Mastercard or American Express account—by mail include credit card name, number and expiration date, or if you prefer, use our 24-hour toll-free number, 800-631-8112. In N.J. only 201-540-0445.

*N.J. Residents add 5% tax.

creative computing

REMEMBER:



"NEVER FORGETS."

MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

Distributed Exclusively by **Leading Edge Products, Inc.**, 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.